DTIC FILE COPY

RADC-TR-89-324
In-House Report
November 1989

AD-A218 122

INTEGRATION OF THE SKY COMPUTERS VORTEX/AT ARRAY PROCESSOR INTO THE WORKSTATION COMMUNICATIONS SIMULATOR

Daniel R. Kupiak

DTIC
ELECTE
FEB 13 1990
S E D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

90 02 12 200

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC TR-89-324 has been reviewed and is approved for publication.

APPROVED:

Thadeus J. Domurat

THADEUS J. DOMURAT
Chief, Signal Intelligence Division
Directorate of Intelligence and Reconnaissance

APPROVED:

Walter J. Senus

WALTER J. SENUS
Technical Director
Directorate of Intelligence and Reconnaissance

FOR THE COMMANDER:

Igor G. Plonisch

IGOR G. PLONISCH
Directorate of Plans and Programs

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRAP) Griffiss AFB NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) RADC-TR-89-324	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Rome Air Development Center 6b. OFFICE SYMBOL (If applicable) IRAP	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (IRAP)
6c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	7b. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) IRAP	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State, and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 35885G; PROJECT NO.: 685L; TASK NO.: PR; WORK UNIT ACCESSION NO.: OJ

11. TITLE (Include Security Classification)
INTEGRATION OF THE SKY COMPUTERS VORTEX/AT ARRAY PROCESSOR INTO THE WORKSTATION COMMUNICATIONS SIMULATOR

12. PERSONAL AUTHOR(S)
Daniel R. Kupiak

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
In-House	FROM Jan 88 TO Jun 88	November 1989	126

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Digital Communications; Modulation; Demodulation; Simulation Encoding; Decoding; Simulators; Computerized Simulation; Data Transmission Systems.
25	05		
09	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Workstation Communications Simulator (WCS) is a personal computer based hardware/software system for the interactive digital simulation of point-to-point digital communications systems. The baseline WCS is hosted on an IBM PC/AT microprocessor and is written entirely in FORTRAN-77. The enhanced WCS employs an additional optional single-board floating point array processor to perform the computationally intensive signal processing calculations.

This report describes the initial implementation of the enhanced WCS utilizing a VORTEX/AT array processor manufactured by Sky Computers, Inc. The computational throughput gains experienced over the baseline WCS are consistent with expectations and are on the order of a factor of three. The throughput gain would have been considerably higher had it not been for the fact that the microcoded Gaussian random number generator did not function correctly and was not used. Further increases in computational throughput can be obtained through the optimization of the enhanced WCS software for use with the VORTEX/AT.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL DANIEL R. KUPIAK	22b. TELEPHONE (Include Area Code) (315) 330-3206 22c. OFFICE SYMBOL RADC(IRAP)

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

ACKNOWLEDGEMENT

This report was submitted in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering from Rensselaer Polytechnic Institute.

The author would like to gratefully acknowledge the advice and assistance of Dr. James Modestino, the project advisor, and Dr. Kurt Matis for their support in the preparation of this report.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 8

CONTENTS

LIST OF TABLES v

LIST OF FIGURES vii

LIST OF ACRONYMS ix

ABSTRACT xi

1 INTRODUCTION AND HISTORICAL REVIEW 1

2 BASELINE WCS 5
2.1 SYSTEM MODEL 5
2.1.1 Binary Message Source 8
2.1.2 Channel Encoder 9
2.1.3 Modulator/Transmitter 10
2.1.4 Channel 16
2.1.5 Demodulator/Receiver 21
2.1.6 Channel Decoder 23
2.1.7 Remote Destination 23
2.2 FUNCTIONAL DESCRIPTION 24
2.2.1 DESIGN Mode 24
2.2.2 VALIDATION Mode 26
2.2.3 SIMULATION Mode 26
2.2.4 ANALYSIS Mode 27

3 VORTEX/AT ARRAY PROCESSOR 29
3.1 Hardware 29
3.2 Architecture 31
3.3 Software 33
3.3.1 VEX Preprocessor 33
3.3.2 Direct Command Interface 34
3.3.3 Custom Microcode Development 35

4 ENHANCED WCS 37
4.1 RANDOM NUMBER GENERATION 37
4.1.1 Uniform Random Number Generator 37
4.1.2 Gaussian Random Number Generator 41
4.2 BINARY SOURCE MODULES 51
4.2.1 Equiprobable Binary Source 51

4.2.2 All Ones Binary Source 56
4.2.3 All Zeros Binary Source 56
4.3 MODULATOR/TRANSMITTER MODULES 56
4.3.1 BPSK Modulator/Transmitter 57
4.3.2 DPSK Modulator/Transmitter 58
4.3.3 QPSK Modulator/Transmitter 59
4.3.4 OQPSK Modulator/Transmitter 60
4.4 CHANNEL MODULES 61
4.4.1 AWGN Channel 61
4.4.2 Inverting Channel 63
4.5 DEMODULATOR/RECEIVER MODULES 64
4.5.1 BPSK Demodulator/Receiver 64
4.5.2 DPSK Demodulator/Receiver 65
4.5.3 QPSK Demodulator/Receiver 67
4.5.4 OQPSK Demodulator/Receiver 69
4.6 BINARY SINK MODULES 71
4.6.1 Equiprobable Binary Sink 71
4.6.2 All Ones Binary Sink 71
4.6.3 All Zeros Binary Sink 72
4.7 VORTEX/AT SUPPORT MODULES 72
4.7.1 VORTEX/AT Initialize 72
4.7.2 VORTEX/AT Release 73
4.8 MISCELLANEOUS SUPPORT MODULES 73
4.8.1 Date 73
4.8.2 Time of Day 73
4.8.3 Timer 74
4.8.4 Q Function 74

5 ENHANCED WCS PERFORMANCE **77**
5.1 Enhanced WCS Subset Performance 77
5.2 Enhanced WCS Performance 96
5.3 Enhanced WCS Timing Estimate 101

6 DISCUSSION AND CONCLUSIONS **103**

LITERATURE CITED **107**

LIST OF TABLES

2.1 Typical Modulation Capabilities Available in the WCS 17

4.1 Statistics For Generated Gaussian Distributions 47

5.1 Numerical Results for BPSK, without VORTEX/AT 80

5.2 Numerical Results for BPSK, with VORTEX/AT 82

5.3 Numerical Results for DPSK, without VORTEX/AT 84

5.4 Numerical Results for DPSK, with VORTEX/AT 86

5.5 Numerical Results for QPSK, without VORTEX/AT 88

5.6 Numerical Results for QPSK, with VORTEX/AT 90

5.7 Numerical Results for OQPSK, without VORTEX/AT 92

5.8 Numerical Results for OQPSK, with VORTEX/AT 94

5.9 Simulation Times, 50,000 Input Data Bits, 1600 Bit Processing Blocks 102

5.10 Simulation Times, 50,000 Input Data Bits, 1024 Bit Processing Blocks 102

LIST OF FIGURES

2.1 General Model for a Communication System 6

2.2 Fading Multipath Channel Model 19

2.3 WCS Organization 25

3.1 VORTEX/AT 30

3.2 VORTEX/AT Architecture 32

4.1 Empirical Cumulative Distribution Function, Original Algorithm . . . 39

4.2 Empirical Cumulative Distribution Function, New Algorithm 40

4.3 Empirical Cumulative Distribution Function, VORTEX/AT 42

4.4 Empirical Cumulative Distribution Function, Baseline WCS Gaussian Random Number Generator 44

4.5 Empirical Cumulative Distribution Function, Enhanced WCS Gaussian Random Number Generator 45

4.6 Empirical Cumulative Distribution Function, VORTEX/AT Microcode Gaussian Random Number Generator 46

4.7 P–P Plot, Baseline WCS Gaussian Random Number Generator . . . 48

4.8 P–P Plot, Enhanced WCS Gaussian Random Number Generator . . . 49

4.9 P–P Plot, VORTEX/AT Microcode Gaussian Random Number Generator 50

4.10 Q–Q Plot, Baseline WCS Gaussian Random Number Generator . . . 52

4.11 Q–Q Plot, Enhanced WCS Gaussian Random Number Generator . . 53

4.12 Q–Q Plot, VORTEX/AT Microcode Gaussian Random Number Generator 54

4.13 Q–Q Plot, VORTEX/AT Microcode Gaussian Random Number Generator vs Gaussian with 1.67 Standard Deviation 55

5.1 Block Diagram of Enhanced WCS Subset 78

5.2 Probability of Bit Error for BPSK, without VORTEX/AT 81

5.3 Probability of Bit Error for BPSK, with VORTEX/AT 83

5.4 Probability of Bit Error for DPSK, without VORTEX/AT 85

5.5 Probability of Bit Error for DPSK, with VORTEX/AT 87

5.6 Probability of Bit Error for QPSK, without VORTEX/AT 89

5.7 Probability of Bit Error for QPSK, with VORTEX/AT 91

5.8 Probability of Bit Error for OQPSK, without VORTEX/AT 93

5.9 Probability of Bit Error for OQPSK, with VORTEX/AT 95

5.10 Probability of Bit Error for BPSK 97

5.11 Probability of Bit Error for DPSK 98

5.12 Probability of Bit Error for QPSK 99

5.13 Probability of Bit Error for OQPSK 100

LIST OF ACRONYMS

AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
CPU	Central Processing Unit
CRT	Cathode Ray Terminal
DPSK	Differential Phase Shift Keying
DCI	Direct Command Interface
DMA	Direct Memory Access
IBM	International Business Machines
ICS	Interactive Communications Simulator
ISI	InterSymbol Interference
FLOP	Floating Point OPeration
MFLOPS	Million Floating Point OPerations per Second
MS-DOS	MicroSoft-Disk Operating System
OQPSK	Offset Quadrature Phase Shift Keying
PC-DOS	Personal Computer-Disk Operating System
PC/AT	Personal Computer/Advanced Technology
PIO	Programmable Input/Output
QPSK	Quadrature Phase Shift Keying
RADC	Rome Air Development Center

RAM	**Random Access Memory**
RF	**Radio Frequency**
WCS	**Workstation Communications Simulator**
WGN	**White Gaussian Noise**
WSSUS	**Wide-Sense Stationary Uncorrelated Scattering**
ZMNL	**Zero Memory NonLinear**

ABSTRACT

The Workstation Communications Simulator (WCS) is a personal computer based hardware/software system for the interactive digital simulation of point-to-point digital communications systems. The baseline WCS is hosted on an IBM PC/AT microprocessor and is written entirely in Fortran-77. The enhanced WCS employs an additional optional single-board floating point array processor to perform the computationally intensive signal processing calculations.

This report describes the initial implementation of the enhanced WCS utilizing a VORTEX/AT array processor manufactured by Sky Computers Inc. The computational throughput gains experienced over the baseline WCS are consistent with expectations and are on the order of a factor of three. The throughput gain would have been considerably higher had it not been for the fact that the microcoded Gaussian random number generator did not function correctly and was not used. Further increases in computational throughput can be obtained through the optimization of the enhanced WCS software for use with the VORTEX/AT.

SECTION 1

INTRODUCTION AND HISTORICAL REVIEW

The Workstation Communications Simulator (WCS) is an IBM PC/AT based hardware/software system designed to simulate the operation of digital point-to-point communication systems. The WCS was developed to meet the needs of a wide variety of users including system developers, communication researchers, and educators desiring a cost-effective, high-performance simulation tool.

The WCS is essentially a modern low-cost version of another communications simulator called the Interactive Communications Simulator (ICS). The ICS was developed at RPI during the late 1970's under the support of the U.S. Air Force Rome Air Development Center (RADC)[1]. The result of this development was a fairly comprehensive system for the digital simulation of a wide variety of point-to-point digital communication systems. The ICS was a flexible, graphics oriented, highly interactive hardware/software system consisting of a Digital Equipment Corporation PDP-11/40 minicomputer acting as a host to a Floating Point Systems AP-120B fast peripheral array processor. The host computer provided the overall system control and user interface while the array processor performed all of the signal processing computations.

The ICS was subsequently significantly enhanced during the early 1980's by PAR Technology Corporation under contract to RADC[2]. While the enhanced version of the ICS was again developed for implementation on a PDP-11/40 host computer

with an AP-120B peripheral array processor, a second version was developed which used a Digital Equipment Corporation VAX-11 series computer as the host. The enhanced ICS, in addition to other improvements, added such extensions as spread-spectrum communications techniques and electronic counter measures (jamming) signals to the original ICS. With the development of the enhanced ICS an extremely powerful communication system development and training tool was available.

The major shortcoming of the enhanced ICS, hereafter referred to as just ICS, was initially the relatively high cost of the hardware involved. This fact limited the widespread acceptance of the ICS as an educational or research tool. Second, as the AP-120B was required for all implementations of the ICS, all signal processing software modules had to be written in AP-120B assembly language. This hindered the development of new modules which, through evolution, were found to be useful additions to the original ICS repertoire. In addition, as time passed, the technology used in the AP-120B array processor rapidly became obsolete. These factors led to the development of the WCS[3].

The WCS was developed using an IBM PC/AT computer as the host. The initial version of the WCS, the baseline WCS, did not require any peripheral array processor to perform the signal processing computations. This version was implemented entirely in host computer FORTRAN-77. The standardized hardware and software environment of the baseline WCS allowed the development of additional software modules to be performed quite easily.

The development of single-board memory-mapped array processors for floating-point computation enabled the development of an enhanced WCS. These array processors offered the potential of increasing the throughput of the WCS to as high as 20 million floating point operations[1] per second (MFLOPS). This was a considerable improvement over the approximately 12 MFLOPS rate of the VAX-11 based ICS.

The objective of this project was to enhance the performance of the baseline WCS through the implementation of new signal processing modules which could take advantage of the throughput gains possible through the use of an array processor. The Sky Computers VORTEX/AT[4] single-board array processor was chosen for this effort.

[1] A floating point operation (FLOP) consists of a floating point multiplication or addition.

SECTION 2

BASELINE WCS

The baseline WCS has proven to be a flexible and useful tool in analyzing communication system performance. The general philosophy in developing the WCS was that simulation should be viewed as an adjunct to, rather than a substitute for, analysis. While the basic premise of the WCS is Monte-Carlo simulation, every effort was made to incorporate analytical and graphical techniques to enhance and extend the capabilities of the system.

2.1 SYSTEM MODEL

A block diagram of the general model for a communication system is shown in Fig. 2.1. In general, the information to be conveyed to the destination consists of a sequence of elements called *messages*. The *message set* is a set of M elements that are indexed by the integers $0, 1, 2, \ldots, M-1$, and represents all possible messages.

The information source generally produces either a discrete sequence or an analog waveform which is encoded, transmitted over a specified channel, reconstructed, and delivered to an information sink. The purpose of the source encoder is to encode the source output, either digital or analog, into a discrete sequence for subsequent channel encoding and transmission. The source decoder provides the inverse function and attempts to reconstruct an approximation of the source output at the destination. Since the primary purpose of the WCS is the evaluation of the digital

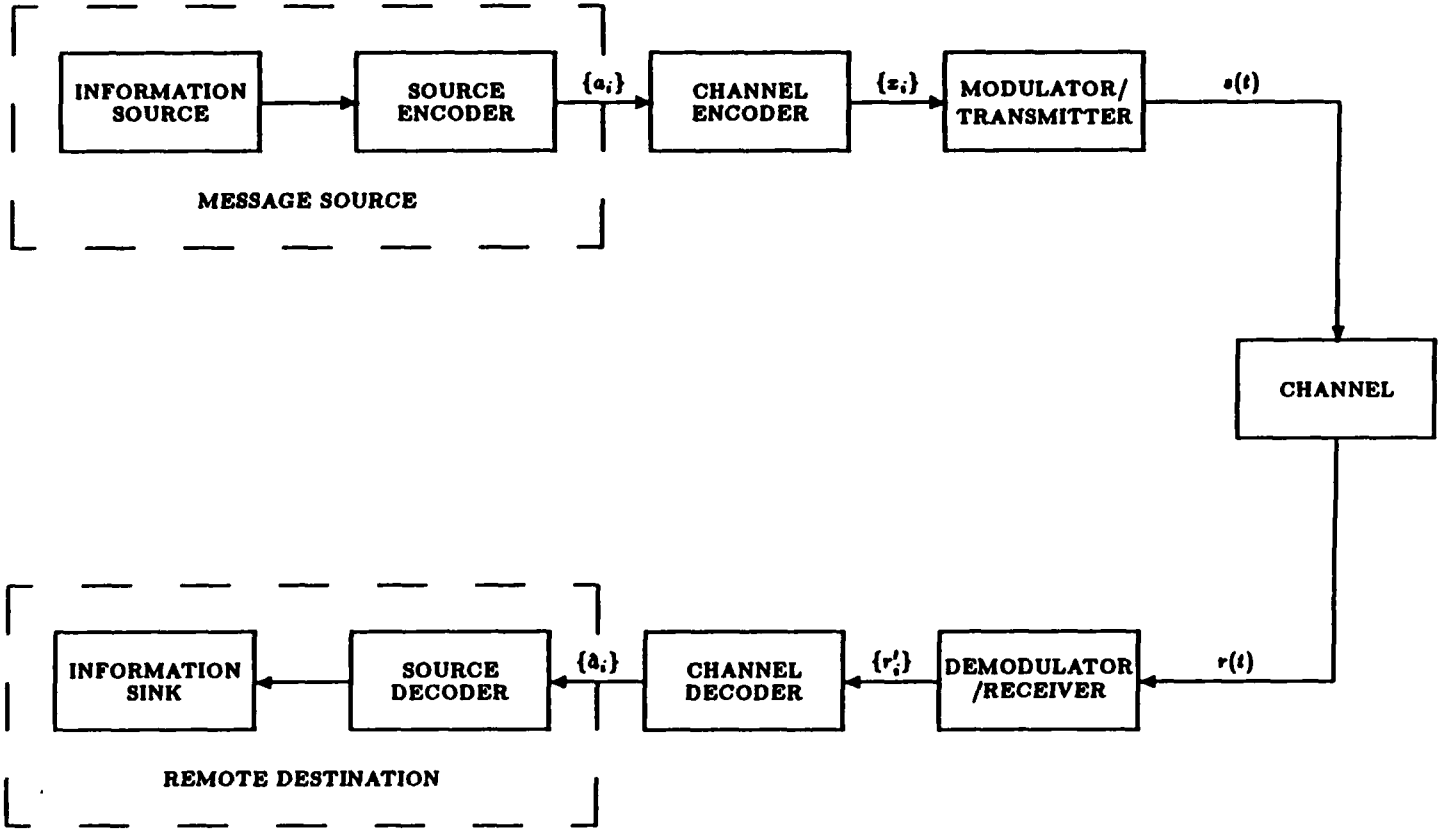

Figure 2.1: General Model For a Communication System

data channel, source coding/decoding have not been incorporated into the simulation. As a result, the information source and source encoder are combined into a single element called the message source. The information sink and source decoder have similarly been combined into a single element called the remote destination.

The channel encoder introduces, in some controlled manner, redundancy in the message sequence for the purpose of controlling the detrimental effects of noise and interference in the channel. The modulator/transmitter produces signals that are used to represent the messages on the channel. During each transmission interval, the message source produces one of the M messages, and the corresponding signaling waveform is transmitted over the channel to the demodulator/receiver.

Rather than attempting to reproduce the transmitted waveform (as in an analog communication system), the goal of the demodulator/receiver in a digital communication system is to determine which of the M messages was sent. The demodulator/receiver first produces a real number, or vector, called the *decision statistic.* This statistic is then input to a decision device which decides which message was sent based upon a predetermined strategy referred to as a *decision rule.* Decision rules are selected according to various criteria such as the average probability of error.

The set of allowable decisions can be different from the message set such as when the receiver is allowed to erase symbols. For example, the binary erasure channel has a message set $\{0, 1\}$, but the set of allowable decisions is $\{0, 1, e\}$, where e denotes

an erasure. This is in contrast to the binary *hard decision channel* which has the set $\{0, 1\}$ as both the message set and the set of allowable decisions.

The purpose of the channel decoder is to use the redundancy introduced by the channel encoder to try to correct any errors that may have occurred in transmission. The measure used to determine how well the digital data channel performs is the frequency with which errors occur in the decoded sequence. In general, the probability of error is a function of the code characteristics, the type of waveform used to transmit the information over the channel, the characteristics of the channel, and the method of demodulation and decoding.

2.1.1 Binary Message Source

The message set used in the WCS is constrained to the case where $M = 2$, i.e., binary, and therefore the message source is called a *binary message source.* In addition, the message source is also assumed to be memoryless. This binary memoryless source is modeled by the pseudo-randomly generated binary independent and identically distributed (i.i.d) sequence $\{a_i\}$ as indicated in Fig 2.1.

The binary memoryless source is implemented as the modulo-2 sum of a linear-feedback shift register of length $2^{47} - 1$ and a mixed congruential random number generator[2,5]. It has a period exceeding $2^{47} - 1$.

2.1.2 Channel Encoder

The purpose of the channel encoder is to accept the binary sequence $\{a_i\}$ as its input and produce the binary (M-ary in the general case) channel symbol sequence $\{x_i\}$ at its output. Although the encoded output sequence could consist of $x_i \in 0, 1$, it is more convenient to use an *antipodal* channel symbol set where $x_i \in -1, 1$.

The WCS provides the capability to perform both *block* and *convolutional* channel encoding. When block encoding is used, the binary sequence $\{a_i\}$ is segmented into blocks of length k to which $n - k$, redundant bits are added to produce a (n, k) block code with a rate $R = k/n$, measured in units of information bits per transmitted channel symbol. There is complete independence between the output blocks. The WCS includes several Hamming, Golay, and Bose-Chaudhuri-Hocquenghem (BCH) block codes. When convolutional encoding is used the binary sequence $\{a_i\}$ is broken into segments of length b, called information frames, which may be as short as one bit. The encoder can store K of these frames at one time. During each frame time, a new information frame is shifted into the encoder, and the oldest previous information frame is shifted out and discarded. At the end of any frame time, the encoder has stored the most recent K information frames, for a total of Kb information input bits. The encoder then computes a single output codeword of length n, using n linear algebraic function generators (typically modulo-2 adders). This codeword is shifted out of the encoder as the next information frame is shifted in. Hence, the channel must transmit n codeword symbols (bits for the binary case)

for each b information bits. The rate R of the convolutional code is defined as $R = b/n$. The quantity K is defined as the *constraint length* of the code. Note that each input bit affects $[K/b]\,n$[1] encoder output symbols. The WCS includes tabulations of good convolutional code constructions for various values of K and R. In addition, the user can construct a new code merely by specifying the code connection vectors describing the pattern for connecting the shift register stages to the modulo-2 adders.

Existing block and/or convolutional codes are generally effective in correcting random errors but are deficient in their ability to correct burst errors. To limit the effect of burst errors, various interleaving schemes are used. In particular, the channel symbol sequence $\{x_i\}$ is scrambled in such a manner that the successive channel symbols are transmitted separated by many channel signaling intervals. The interleaver implemented in the WCS is of a convolutional type with a maximum symbol separation of 30. It is a particular implementation of a Type I optimal interleaver as described by Ramsey[6].

2.1.3 Modulator/Transmitter

The purpose of the modulator/transmitter is to map the channel symbol sequence $\{x_i\}$ (possibly interleaved) into a waveform $s(t)$ suitable for transmission over the channel. Channel symbols are presented to the modulator/transmitter ev-

[1]The notation $[x]$ means the largest integer not exceeding x.

ery T_s seconds producing a time-limited channel signaling waveform which vanishes outside of an interval of duration T_s seconds. The quantity T_s is called the *baud interval*, or equivalently, $f_s = 1/T_s$ is called the *baud rate.*

It is convenient to normalize all frequency (time) domain quantities to the baud rate (interval). For example, the bandwidths of frequency-selective channel filtering elements are all normalized to f_s. The filter cutoff, or 3 dB point, would then be specified in digital frequency units of cycles per baud. This allows the user to configure and execute a simulation program independent of the actual baud rate and provides simulation results in a convenient parametric form.

In simulating the channel signaling waveform, $s(t)$, complex notation is extensively used. This allows the simulation of a bandpass signal in terms of its lowpass complex envelope and eliminates the need for simulating a radio-frequency (RF) carrier component.

In general, an RF signal can be considered to be of the form

$$s(t) = Rr(t)\cos\left[2\pi f_c t + \theta(t) + \phi\right] \tag{2.1}$$

where $r(t)$ represents amplitude modulation and $\theta(t)$ represents phase modulation. The variable R is a real number that represents the signal amplitude at the receiver. Physically the parameter R may be the peak signal voltage, such as when $|r(t) = 1|$. The variable ϕ is the phase of the RF carrier at the time $t = 0$ in the absence of any phase modulation (i.e. $\theta(t) = 0$). This phase angle may or may not be known to the receiver. The variable f_c is the *assumed known* carrier frequency.

Equation 2.1 can be further simplified if the initial phase at time $t = 0$ is absorbed into $\theta(t)$ as follows

$$s(t) = Rr(t)\cos\left[2\pi f_c t + \theta(t)\right]. \tag{2.2}$$

Similarly, the value R can be absorbed into $r(t)$ resulting in

$$s(t) = r(t)\cos\left[2\pi f_c t + \theta(t)\right]. \tag{2.3}$$

Using the trigonometric identity

$$\cos(A \pm B) = \cos A \cos B \mp \sin A \sin B$$

equation 2.3 may be rewritten as

$$\begin{aligned} s(t) &= r(t)\left[\cos 2\pi f_c \cos\theta(t) - \sin 2\pi f_c \sin\theta(t)\right] && (2.4)\\ &= r(t)\cos 2\pi f_c \cos\theta(t) - r(t)\sin 2\pi f_c \sin\theta(t) && (2.5) \end{aligned}$$

If the quantities $s_c(t)$ and $s_s(t)$ are now defined such that

$$\begin{aligned} s_c(t) &= r(t)\cos\theta(t) && (2.6)\\ s_s(t) &= r(t)\sin\theta(t) && (2.7) \end{aligned}$$

then equation 2.5 can be rewritten as

$$s(t) = s_c(t)\cos 2\pi f_c t - s_s(t)\sin 2\pi f_c t. \tag{2.8}$$

This is known as the *quadrature-carrier* description of $s(t)$, where $s_c(t)$ is called the *in-phase* component and $s_s(t)$ is called the *quadrature* component of $s(t)$. If the

complex envelope $\tilde{s}(t)$ of $s(t)$ is now defined as

$$\begin{aligned} \tilde{s}(t) &= s_c(t) + j s_s(t) && (2.9) \\ &= r(t)\cos\theta(t) + j r(t)\sin\theta(t) \\ &= r(t)\left[\cos\theta(t) + j\sin\theta(t)\right] \\ &= r(t)e^{j\theta(t)}. && (2.10) \end{aligned}$$

then equation 2.3 can be rewritten as

$$\begin{aligned} s(t) &= r(t)\cos\left[2\pi f_c t + \theta(t)\right] \\ &= \mathrm{Re}\left\{r(t)\cos\left[2\pi f_c t + \theta(t)\right] + j r(t)\sin\left[2\pi f_c t + \theta(t)\right]\right\} \\ &= \mathrm{Re}\left\{r(t)e^{\left[j2\pi f_c t + \theta(t)\right]}\right\} \\ &= \mathrm{Re}\left\{r(t)e^{j2\pi f_c t}e^{j\theta(t)}\right\} \\ &= \mathrm{Re}\left\{\tilde{s}(t)e^{j2\pi f_c t}\right\} && (2.11) \\ &= \frac{1}{2}\tilde{s}(t)e^{j2\pi f_c t} + \frac{1}{2}\tilde{s}^*(t)e^{-j2\pi f_c t} && (2.12) \end{aligned}$$

Equations 2.11 and 2.12 are known as the complex envelope description of $s(t)$. By generating only the real-valued in-phase and quadrature (I/Q) components, the simulator need not be concerned with the explicit details of the RF portions of the communication system. In particular, discrete samples of the I/Q components are generated at equally spaced intervals which are determined through a user specified sampling rate.

If the assumption is made that $f_c t \gg 1$ and $r(t)$ and $\theta(t)$ are baseband waveforms of duration T_s then the signal energy values may be given as shown below. Th

energy per data pulse given that the kth message is being sent is

$$E_k = \int_0^{T_s} s^2(t)\,dt. \tag{2.13}$$

If $\theta(t)$ is equal to a constant θ_k for $0 \leq t < T_s$ if the kth message is sent during the interval $[0, T_s]$, then using equation 2.2,

$$E_k = \left(R^2/2\right) \int_0^{T_s} r^2(t)\,dt. \tag{2.14}$$

In the WCS, this has resulted in the constraint

$$\int_{(i-1)T_s}^{iT_s} |\tilde{s}(t)|^2 dt = E_s; \qquad i = 1, 2, \ldots, \tag{2.15}$$

being applied where E_s represents the constant signal energy per channel signaling element or baud.

Since the simulation is performed digitally equation 2.15 is actually written as

$$\sum_{k=N(i-1)}^{Ni} |\tilde{s}(k\Delta T_s)|^2 = E_s; \qquad i = 1, 2, \ldots \tag{2.16}$$

where N is the number of samples per baud and is user selectable. Typical values of N are $N = 8, 16$, or 32. The quantity $\Delta T_s = T_s/N$ is the *sampling period.*

For the purpose of simulation, it is sufficient to express the complex envelope of the signaling waveform in the general form

$$\tilde{s}(t) = \sqrt{\frac{2E_s}{T_s}} \sum \left[C_{c_i} u_c(t - iT_s - \tau) + jC_{s_i} u_s(t - iT_s - \tau)\right] e^{j\theta(t)} \tag{2.17}$$

where τ is the random timing epoch and $\theta(t)$ is the carrier phase. The receiver will be required to estimate and track these initially unknown quantities. The sequences $\{C_{c_i}\}$ and $\{C_{s_i}\}$ are determined from the channel symbol sequence $\{x_i\}$

while $u_c(t)$ and $u_s(t)$ are elementary baseband signaling waveforms. Through appropriate choices for the mappings $\{C_{c_i}\}$ and $\{C_{s_i}\}$, together with the baseband signaling waveforms $u_c(t)$ and $u_s(t)$, equation 2.17 is sufficiently general to include most typical modulation formats. These include: coherent binary phase-shift keying (BPSK), differentially coherent phase-shift keying (DPSK), quadrature phase-shift keying (QPSK), noncoherent frequency-shift keying (FSK), and quadrature amplitude-shift keying (QASK). Equation 2.17 must be modified slightly to include the case of offset or staggerred quadrature phase-shift keying (OQPSK) and the constant envelope formats such as continuous phase frequency-shift keying (CPFSK) and the special case of minimum-shift keying (MSK). Specifically, in this case

$$\tilde{s}(t) = \sqrt{\frac{2E_s}{T_s}} \sum [C_{c_i} u_c(t - 2iT_s - \tau) + jC_{s_i} u_s(t - 2iT_s - \tau)]\, e^{j\theta(t)} \qquad (2.18)$$

where now the elementary baseband waveforms are translated by two baud intervals at a time. The WCS will then generate the lowpass I/Q components $\{s_c(k\Delta T_s)\}$ and $\{s_s(k\Delta T_s)\}$.

For example, in the case of coherent BPSK, $\{C_{c_i}\} = \{x_i\}$, $\{C_{s_i}\} = 0$, $u_s(t) = 0$, and $u_c(t)$ is equal to the pulse-like waveform

$$u_0(t) = \begin{cases} 1; & 0 \le t \le T_s \\ 0; & \text{elsewhere} \end{cases}$$

The lowpass I/Q samples are then generated as

$$\begin{aligned} \{s_c(k\Delta T_s)\} &= x_i \sqrt{\frac{2E_s}{T_s}} \cos\theta; \qquad (i-1)N \le k < iN, \\ \{s_s(k\Delta T_s)\} &= 0; \qquad\qquad\qquad (i-1)N \le k < iN \end{aligned}$$

where $i = 1, 2, \ldots,$ is the baud interval.

Descriptions of the equations used to simulate all of the modulation types available in the WCS are shown in table 2.1.

2.1.4 Channel

In choosing a channel model for the WCS, it is important to provide a fairly general approach which will encompass the entire range of potential applications. In typical communication channels of interest, there are several propagation paths between the transmitter and the receiver. The signal components arriving at the receiver from different paths add constructively or destructively depending upon the difference in the propagation times and on any phase shifts introduced by such occurrences as reflections from objects. Thus the signal strength depends on the relative phasing of the components of the received signal and gives rise to the condition known as *fading.*

If the differential propagation times for the paths are small compared with the data symbol duration, and the path strengths, propagation times, and phase shifts are nearly constant for the duration of the symbol, then there is no significant dispersion of the signal. In this case the fading is referred to as *slow nonselective fading.* If the differential propagation times are large compared with the data symbol duration, signal dispersion will occur and the channel is known as a *frequency-selective* or *time-dispersive* fading channel.

For channels with slow nonselective *Rician* fading, the received signal is the sum

MODULATION	C_{1_i}	C_{2_i}	$u_c(t)$	$u_s(t)$	REMARKS
BPSK	x_i	0	$u_0(t)$	0	$x_i = \pm 1$
DPSK	$x_i C_{1_{i-1}}$	0	$u_0(t)$	0	$x_i = \pm 1$ C_{1_0} assumed known
BFSK	$e^{j\theta_i}$	$x_i e^{j\theta_i}$	$\cos(\Delta\omega t/2)u_0(t)$	$\sin(\Delta\omega t/2)u_0(t)$	$x_i = \pm 1$ $\Delta\omega = 2\pi k/T_s$ θ_i i.i.d. uniform
QPSK	x_{2i}	x_{2i+1}	$(1/\sqrt{2})u_0(t)$	$(1/\sqrt{2})u_0(t)$	$x_i = \pm 1$
MSK	x_{2i}	x_{2i+1}	$\cos(\pi t/2T_s)u_0'(t)$	$\sin(\pi t/2T_s)u_0'(t-T_s)$	$x_i = \pm 1$ I/Q waveforms repeated every $2T_s$ seconds

NOTES: $u_0(t) = 1; \quad 0 \le t \le T_s$ $\quad = 0; \quad$ elsewhere

$u_0'(t) = 1; \quad -T_s \le t \le T_s$ $\quad = 0; \quad$ elsewhere

Table 2.1: Typical Modulation Capabilities Available in the WCS

of a nonfaded version of the transmitted signal and a slow nonselective Rayleigh [2] distributed (or just Rayleigh) faded version. The difference in propagation times is small enough that the channel is nonselective. The nonfaded component of the received signal is called the *specular* component and the Rayleigh faded component is called the random or *diffuse* component. The specular component may result from a reflected path between the transmitter and the receiver and the diffuse component may result from a large number of reflections.

The channel model used in the WCS consists of a three-component fading multipath channel plus additive noise as illustrated in Fig. 2.2. The fading multipath channel is assumed to consist of a single direct path, a specular multipath component, and a diffuse multipath component. The quantity ζ_s in Fig. 2.2 represents the specular component signal energy while ζ_d represents the diffuse component signal energy. The energies in the three signal components are normalized so that the sum of their energies is 1.0. These parameters are under direct user control. In addition, the user is allowed to adjust the relative phase ϕ of the specular multipath component relative to the direct path. Other parameter choices include the nominal differential delay τ_0 and the differential doppler f_0 associated with the two multipath components. The channel assumes the same differential delay and differential doppler for both of these components. The delay and doppler spreads about these nominal values are determined by the choice of the *scattering function* $\sigma(\tau, f)$

[2]The probability density function of the Rayleigh distribution is $f(x) = \frac{2x}{m_f} e^{-x^2/m_f}$ where m_f denotes the second moment of x.

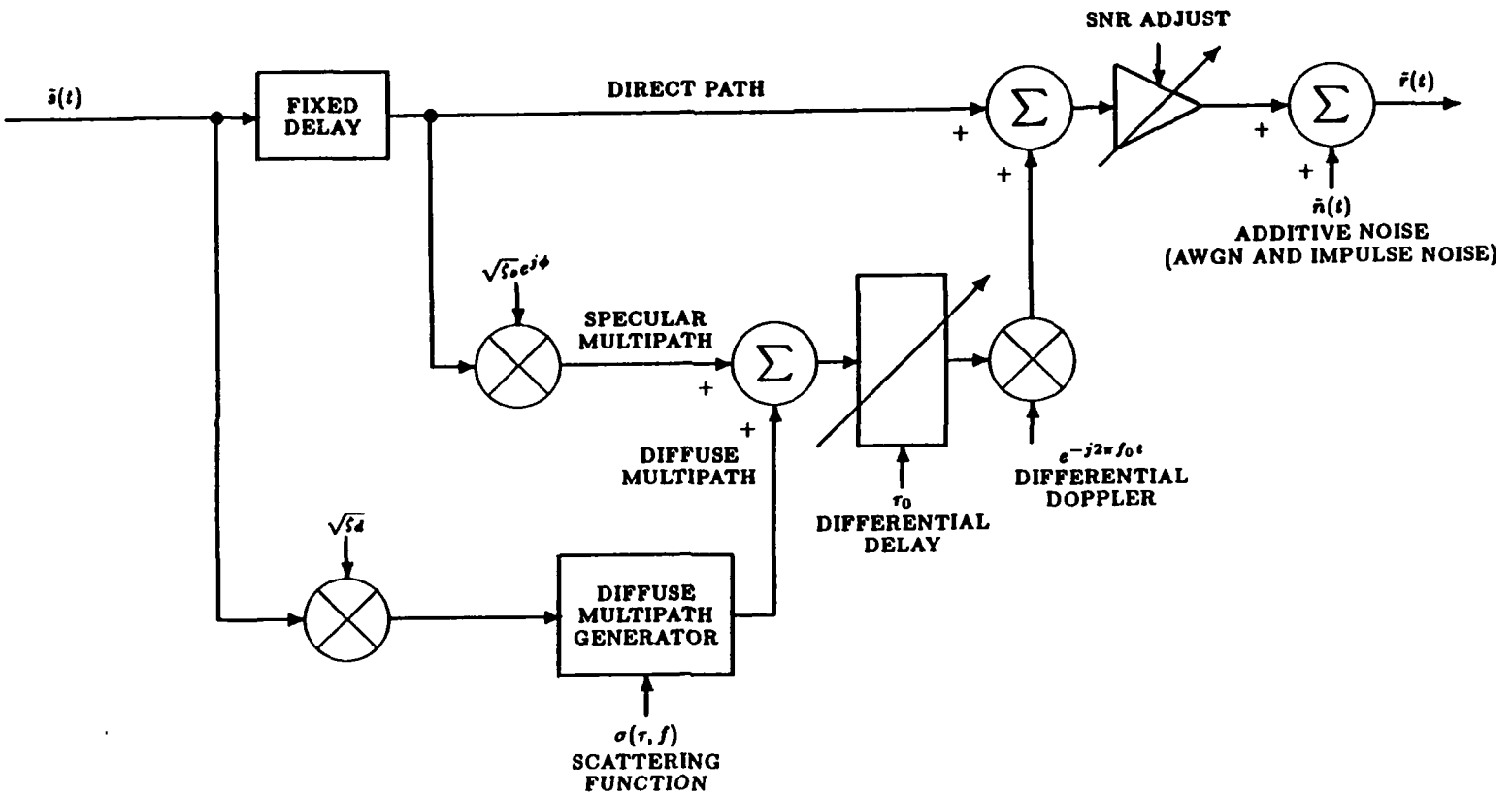

Figure 2.2: Fading Multipath Channel Model

associated with the diffuse multipath component. The quantity $\sigma(\tau, f)$ represents the energy associated with delays in the range $(\tau_0 - d\tau, \tau_0 + d\tau)$ and doppler shifts in the range $(f_0 - df, f_0 + df)$.

The diffuse multipath component has been implemented by means of a tapped delay-line model. This is a fairly general model for fading dispersive channels under the wide-sense stationary uncorrelated scattering (WSSSUS) assumption [7,8].

A general model for the additive noise component of the channel model is the linear combination of a low-density shot noise process and an additive white Gaussian noise (AWGN) component. The complex envelope of the additive noise $\tilde{n}(t)$ can be expressed as

$$\tilde{n}(t) = \tilde{y}(t) + \tilde{w}(t) \tag{2.19}$$

where $\tilde{y}(t)$ is the complex low-density shot noise process and $\tilde{w}(t)$ is the additive white Gaussian noise. The term low-density means that the interarrival times of the impulses are relatively long compared to the typical impulse duration.

The low-density shot noise component can be modeled as the output of a linear time-invariant filter excited by an amplitude modulated impulse train or a point process at its input. The complex envelope of the output of the filter can be expressed as

$$\tilde{y}(t) = y_c(t) + jy_s(t). \tag{2.20}$$

The WCS is capable of simulating shot noise processes where the distribution of the impulse hits ranges from periodic to Poisson distributed.

The complex envelope of the AWGN component can be expressed in terms of its I/Q components according to the equation

$$\tilde{w}(t) = w_c(t) + jw_s(t). \tag{2.21}$$

The components $w_c(t)$ and $w_s(t)$ are modeled as mutually independent zero-mean white Gaussian noise processes, each possessing double-sided noise spectral density $N_0/2$ watts/Hz. Gaussian variates are generated in the WCS by first generating a Rayleigh variate R and a uniform variate θ on the interval $[-\pi, \pi]$. The transformations

$$g_1 = R\sin\theta \tag{2.22}$$

$$g_2 = R\cos\theta \tag{2.23}$$

are then used to produce two independent Gaussian variates. This procedure is much more accurate than approximating the Gaussian distribution with a sum of uniform variates.

2.1.5 Demodulator/Receiver

The main purpose of the demodulator/receiver is to perform data demodulation of the channel output. The signal $r(t)$ is input to this module from the channel and the sequence $\{r_i'\}$ is generated. The demodulator/receiver produces an output once per baud interval.

The demodulator/receiver includes a predetection front end which precedes the data demodulator and/or the other portions of the module. This predetection front

end consists of a *zero-memory nonlinear* (ZMNL) device sandwiched between two narrowband filters. A ZMNL processing capability has been provided in the WCS to allow for the simulation of various unintentional RF saturation effects, such as in a satellite transponder, as well as intentional nonlinear processing, such as might be used to lessen the effects of impulse noise.

In order to perform its basic data demodulation function, the demodulator/receiver module must perform a variety of ancillary functions. These ancillary functions include symbol synchronization, frequency and/or phase tracking, and adaptive symbol equalization. In addition, data symbol deinterleaving is performed within the demodulator/receiver module. The WCS provides the capability to patch symbol synchronization and carrier frequency and/or phase from the transmitter to the receiver. This facility is useful, for example, when the user would like to isolate and eliminate the various synchronization loop tracking dynamics from consideration.

In addition to data demodulators corresponding to each of the modulation strategies included in the WCS, a comprehensive adaptive equalization capability has also been included. More specifically, for the fading multipath channel, the implicit diversity inherent in the multipath spread can be exploited only through adaptive equalization. That is , by resolution of the various multipath components, it is possible to completely remove this source of intersymbol interference (ISI) and, more importantly, realize performance which is an improvement over that obtained in the absence of the multipath. The current trend in high-speed modem design for fading

dispersive channel environments is to exploit this implicit diversity.

2.1.6 Channel Decoder

The purpose of the channel decoder is to accept the sequence $\{r_i'\}$ appearing at the output of the demodulator/receiver and produce the decoded output sequence $\{\hat{a}_i\}$. If convolutional encoding was used in the channel encoder then the Viterbi algorithm[9,10] is employed. If block encoding was used in the channel encoder then several hard decision decoders, using essentially table look-up techniques, are employed. In addition, a fairly general soft decision decoding capability for arbitrary block codes using the Wolf algorithm[11] is provided. The use of this decoding capability requires explicit knowledge of the code parity check matrix. These matrices have been incorporated into the WCS for several selected codes.

2.1.7 Remote Destination

The purpose of the remote destination is to evaluate the overall system performance as measured by the bit error probability $P_b = \Pr\{\hat{a}_i \neq a_i\}$. The reconstructed sequence $\{\hat{a}_i\}$, is compared with a locally generated copy of the message sequence $\{a_i\}$. A modulo-2 addition is performed on the corresponding elements of each sequence and the resulting sum is an indicator of the number of transmission errors.

2.2 FUNCTIONAL DESCRIPTION

The WCS has been designed to be used in four distinct modes of operation as shown in Fig. 2.3. These modes are the DESIGN mode, the VALIDATION mode, the SIMULATION mode, and the ANALYSIS mode. User interaction with the WCS in all modes is through the use of graphics input devices (e.g. crosshairs) and the keyboard.

2.2.1 DESIGN Mode

The DESIGN mode allows the user to assemble a simulation model from the basic functional modules illustrated in Fig. 2.1. Appropriate user aids have been incorporated into the WCS to guide even the casual user through the assembly of a model. Specifically, the block diagram of Fig. 2.1 is displayed on the graphics CRT and the user selects successive blocks to design. When a block is chosen, a menu list appears describing the available options for that functional element. When the user has completed the design of an executable simulation model of a complete end-to-end communication system, the model may be saved as a user-named design file. The user can then exit the DESIGN mode or continue to create alternate executable design files.

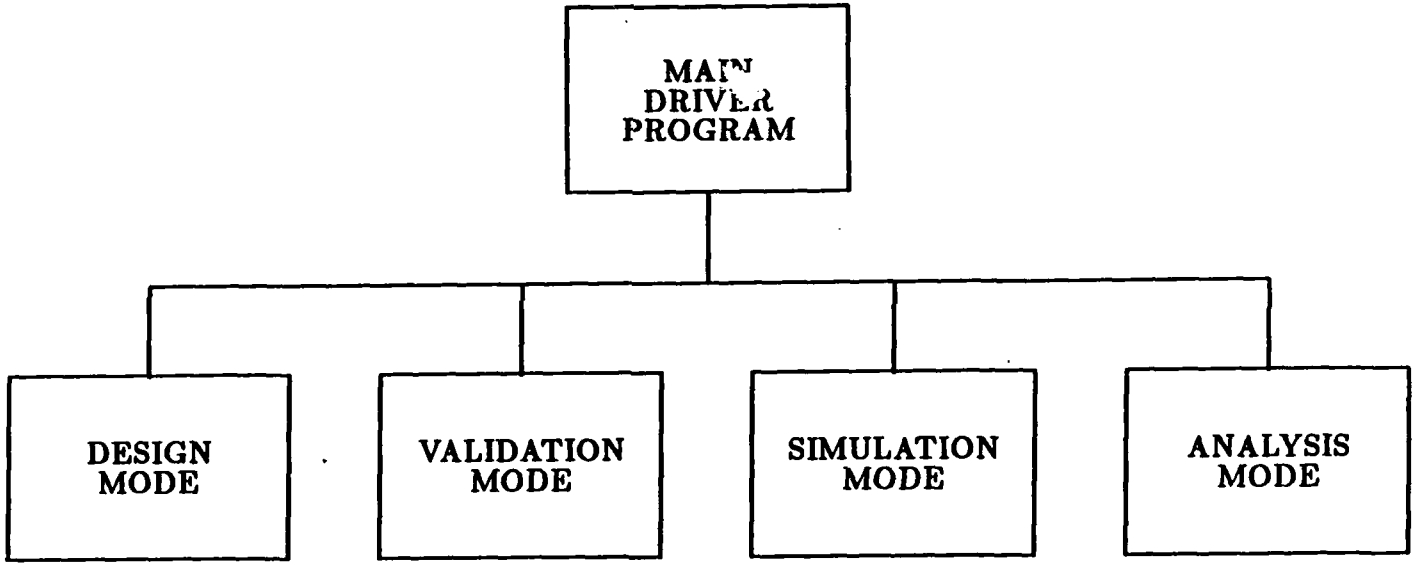

Figure 2.3: WCS Organization

2.2.2 VALIDATION Mode

The VALIDATION mode allows the users to interactively exercise a simulation model which has previously been developed in the DESIGN mode. The purpose of the VALIDATION mode is to verify a simulation model prior to running a time consuming Monte-Carlo simulation. The user is able to view time waveforms and/or frequency domain quantities such as power spectral densities and system transfer functions at various points in the system. This capability allows the user to verify the end-to-end behavior of the simulation model for the communication system under study or the input/output behavior of any specific functional element.

In order to modify or otherwise change a simulation model, the user must leave the VALIDATION mode and reenter the DESIGN mode. Extensive design file editing capabilities have been provided in the WCS to allow the flexible modification of previously developed simulation models.

2.2.3 SIMULATION Mode

The SIMULATION mode performs the actual Monte-Carlo simulation in the WCS. Of primary interest is the evaluation of the bit error probability P_b as a function of the quantity E_b/N_0 where E_b is the signal energy per bit and $N_0/2$ is the double-sided noise density of the Gaussian noise in watts/Hz. The important parameters to be specified by the user are the initial and final values of E_b/N_0 for which simulation results are desired, as well as the increment of E_b/N_0 to be used.

In addition, parameters such as the number of errors to be collected for each value of E_b/N_0 and the maximum number of iterations must also be specified. These latter quantities are necessary in order to ensure the validity of the simulation results. The WCS will then loop through the specified range of E_b/N_0.

In the SIMULATION mode, data logging and bookkeeping software is included to tabulate bit error histories and evaluate bit error probabilities as a function of E_b/N_0. In addition to the straightforward evaluation of bit error counts, other extended error statistics derived from bit error histories are included. These include histograms of the interarrival times between bit errors, the average duration of error bursts, etc. These extended error statistics are useful in providing a more complete description of the communication system performance than just the bit error probability alone.

2.2.4 ANALYSIS Mode

The ANALYSIS mode provides the user with the capability to graphically display the results of a simulation. For example, curves representing the simulated bit error probability P_b as a function of the value E_b/N_0 can be plotted along with the predicted theoretical bounds.

SECTION 3

VORTEX/AT ARRAY PROCESSOR

The Sky Computers Inc. VORTEX/AT is a high speed arithmetic co-processor designed to perform integer and floating point operations at a rate of up to 20 MFLOPS. An IBM PC/AT, or compatible, computer equipped with the VORTEX/AT will perform at a level comparable to that of a minisupercomputer[12], at a fraction of the cost.

The VORTEX/AT performs floating point arithmetic in IEEE-P754 32-bit single and 64-bit double-precision formats, as well as 32-bit 2's complement integer operations and logical functions. The instruction set and object-oriented software architecture are optimized for scalar, vector, and matrix operations.

3.1 Hardware

The VORTEX/AT consists of a single-slot mother-board with two daughter-boards, Fig. 3.1, designed for easy installation into a full-width slot of an IBM PC/AT, or compatible, computer. The current price of the VORTEX/AT is $9,900.

The VORTEX/AT contains one megabyte of on-board dynamic data memory but can address up to one gigabyte. Future versions of the VORTEX/AT will offer up to 8 megabytes of on-board memory. The dynamic data memory is mapped into the PC/AT address space in its entirety at a jumper-selectable location above the 640 KB PC-DOS boundary. In addition to the dynamic data memory, the VORTEX/AT

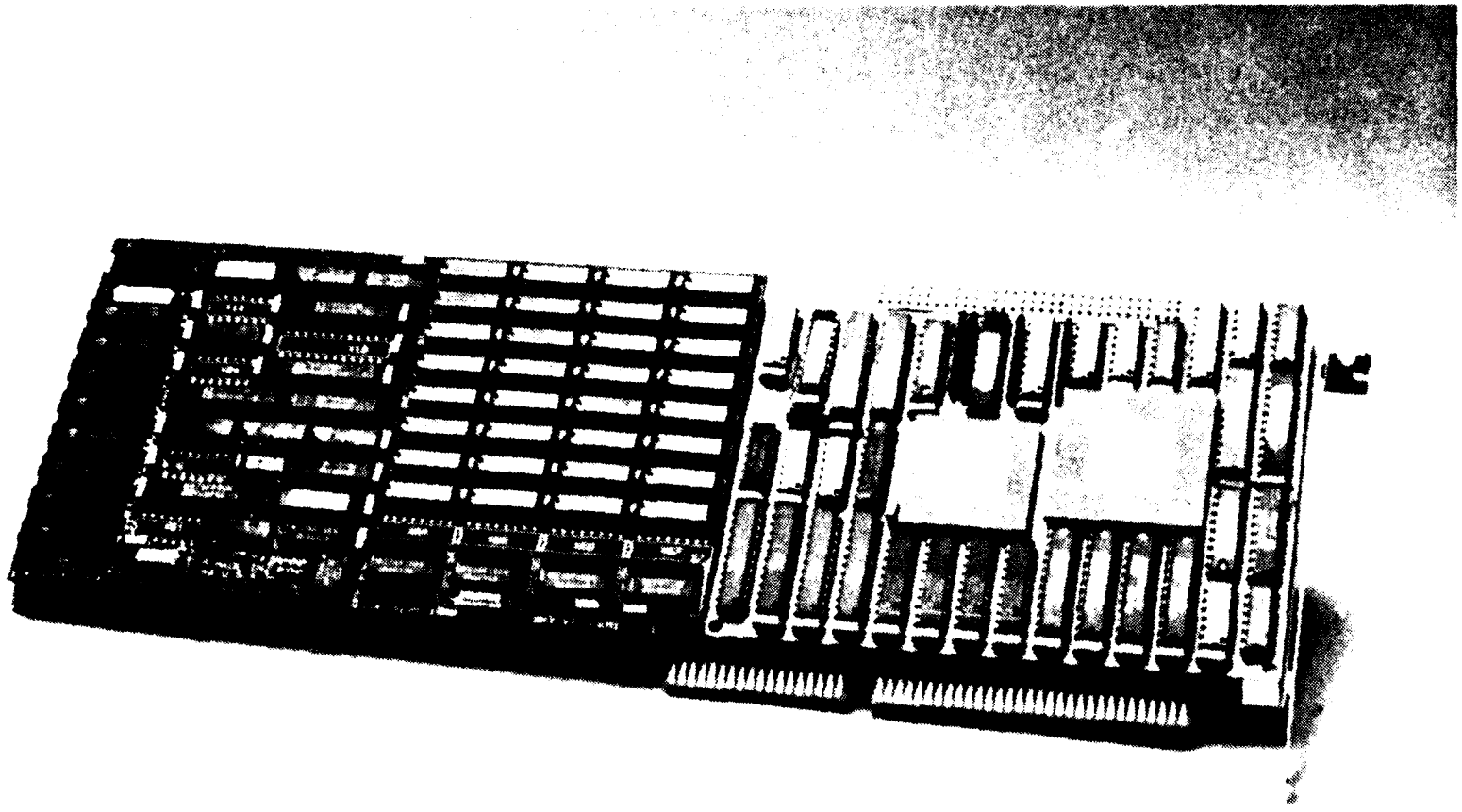

Figure 3.1 VORTEX/AT

also contains 16 kilobytes of fast static data memory and 65 kilobytes of microcode memory.

3.2 Architecture

The block diagram in Fig. 3.2 illustrates the basic architecture of the VORTEX/AT and the interface to the PC/AT host bus. The VORTEX/AT consists of a host bus programmable input/output (PIO) interface to the PC/AT that supports 16-bit data transfers and includes a set of control I/O registers; a control processor that is fully user-programmable, and operates out of a 64 kilobyte program memory; and a large data memory consisting of 1 megabyte of 40 MB/S dynamic ram and 16 kilobytes of 80 MB/S static RAM. An important feature of the VORTEX/AT architecture is the fact that both the host bus interface control I/O registers and the data memory are memory-mapped into the PC/AT 24-bit address space. Consequentely the VORTEX/AT is directly accessible by the host application program without the latency of direct memory access (DMA) transfers.

The only difficulty with this memory-mapped approach is that the 80286 CPU in the PC/AT must be operated in protected memory mode in order to access this shared memory. Protected memory mode operation is not currently supported by either the PC-DOS or MS-DOS operating systems. Therefore, the application must explicitly command the 80286 CPU to enter protected memory mode. This is easily accomplished through the use software library modules provided with the

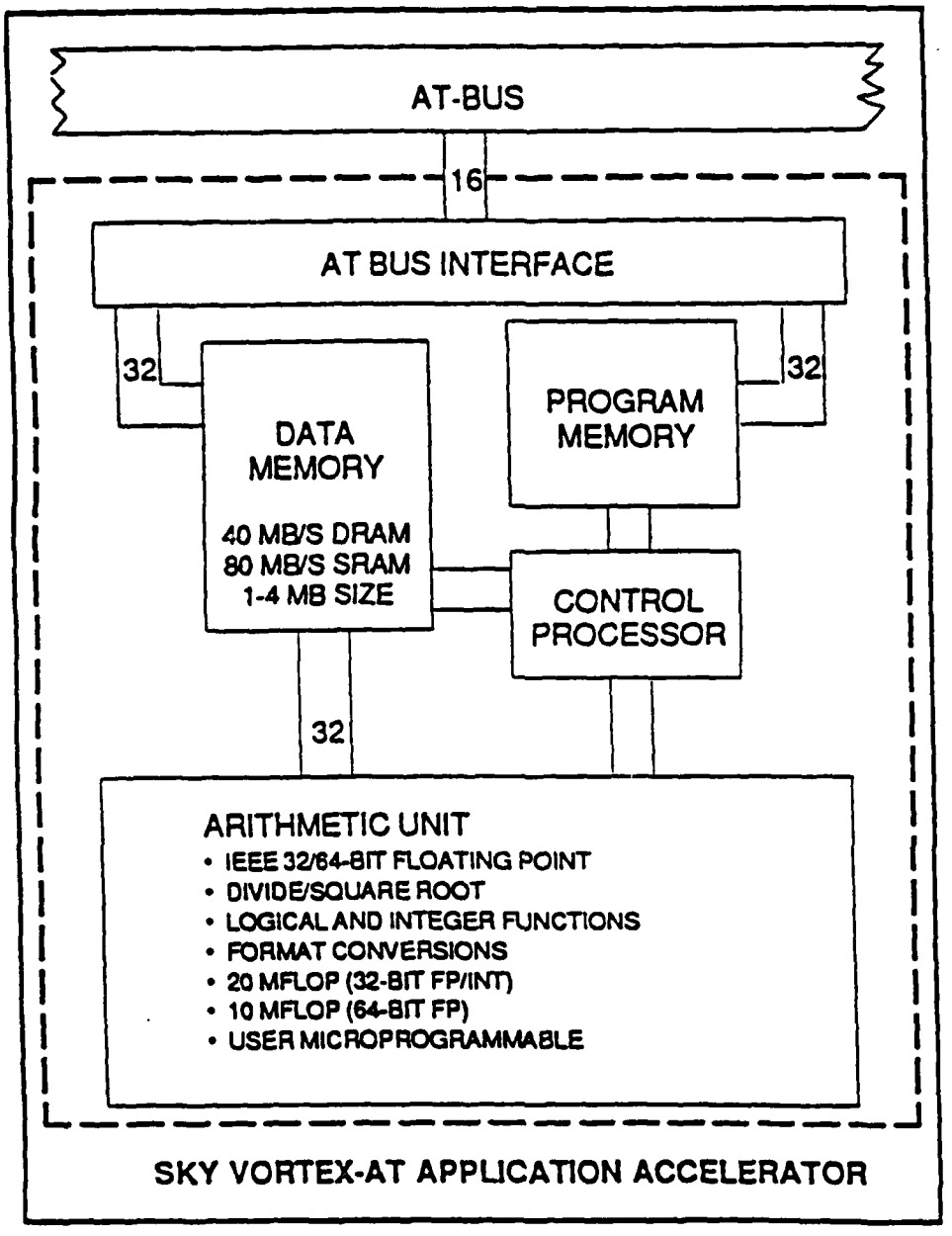

Figure 3.2: VORTEX/AT Architecture

VORTEX/AT.

3.3 Software

The VORTEX/AT programming environment affords a three-level hierarchical approach to application code development[13]. The first level involves a vectorizing preprocessor called VEX, the second level involves the Direct Command Interface, a set of FORTRAN-callable routines that provide high-level language access to the VORTEX/AT, and the third level involves custom microcode development.

3.3.1 VEX Preprocessor

The VEX preprocessor provides the applications programmer with access to the whole range of VORTEX/AT facilities directly from a high-level language. VEX reads a FORTRAN-77 source file as input and translates it into a separate, but equivalent, output file containing instructions for both the host and the VORTEX/AT.

VORTEX/AT functionality can be accessed through the VEX preprocessor at three different levels:

At the first level of access, the programmer can use the VEX preprocessor transparently to translate an existing FORTRAN-77 application to an equivalent one adapted for use with the VORTEX/AT. The resulting output file contains some residual FORTRAN control statements interspersed with calls to elementary VORTEX/AT functions. At this level, no knowledge of VORTEX/AT operations is required beyond how to use the VEX preprocessor to process FORTRAN applica-

tions.

At the second level of access, the programmer familiar with VORTEX/AT operations can rearrange the original source code to include direct references to VORTEX/AT operations. At this level of access, the programmer must have a knowledge of some or all of the VORTEX/AT operations.

At the third level of access, the programmer can use the VEX preprocessor to create new command subroutines that implement frequently-used or time-critical user functions. The function subprogram is translated into a special VORTEX/AT command subroutine that gets loaded into VORTEX/AT memory at the same time as the VORTEX/AT microcode. This level of access represents an economical alternative to custom microcode development, offering almost as much performance enhancement, yet requiring significantly less development effort.

3.3.2 Direct Command Interface

The Direct Command Interface (DCI) is a set of FORTRAN-callable routines that provide high-level language access to the VORTEX/AT. Using the DCI routines the programmer edits a FORTRAN-77 source file to insert VORTEX/AT commands and vectorize its calculations. The resulting file contains valid FORTRAN-77 source code ready for the host's compiler.

3.3.3 Custom Microcode Development

The experienced VORTEX/AT programmer can implement important time-critical functions directly in VORTEX/AT microcode. Custom microcode for the VORTEX/AT is developed using the VORTEX/AT microassembler. This approach eliminates the processing overhead associated with individual VORTEX/AT commands in order to achieve ultimate performance. Once microcoded, the new functions can be used as building blocks in the construction of new library definitions.

SECTION 4

ENHANCED WCS

The general philosophy in developing modules for the enhanced WCS is that all new modules be upward compatible with the baseline WCS and be able to optionally use the VORTEX/AT array processor. The new modules that have been developed are either extensions of existing baseline WCS modules or totally new modules. These modules have all been extensively tested, both as standalone program units, and as modules in the WCS.

4.1 RANDOM NUMBER GENERATION

The random numbers used in the WCS are all derived from uniform random numbers generated by the uniform random number generator modules. The distributions other than the uniform are simply transformations of uniform variates.

4.1.1 Uniform Random Number Generator

The uniform variates required in the WCS are generated using a combination of a mixed congruential generator with a recurrence relation

$$Y_n = (Y_{n-1} \times 65 + 7187) \bmod 2^{15} - 1$$

and a generalized Fibonacci generator with a recurrence relation

$$X_n = (X_{n-5} + X_{n-47}) \bmod 2^{15} - 1$$

to derive a final uniform variate

$$U_n = (Y_n + X_n) \bmod 2^{15} - 1.$$

This algorithm was originally used in the ICS and since that time several papers have been published [14,15] which describe an efficient and portable uniform random number generator. This new algorithm was used to develop a new uniform random number generator module for the WCS. When this module was tested, however, it took longer to generate a sequence of random numbers than the original module, RNG1R.FAP. In addition, the uniform random numbers produced did not exhibit any significant improvement in their statistical properties over those produced using the old module. The empirical cumulative distribution function [16], a qualitative indicator of performance, is plotted in Fig. 4.1 and Fig. 4.2 for 10,470 random numbers generated by the original and new modules, respectively. Based upon these factors, the original uniform random number generator algorithm was not replaced.

The major shortcoming of the uniform random number generator module used in the baseline WCS is that it produces only one uniform random number each time it is called. This results in a considerable amount of time being wasted whenever a sequence of uniform variates is required. The uniform random number generator used in the enhanced WCS addresses this problem by generating all of the uniform variates needed for an operation each time that it is called. To allow this module to be backward compatible with the baseline WCS, only one uniform variate can

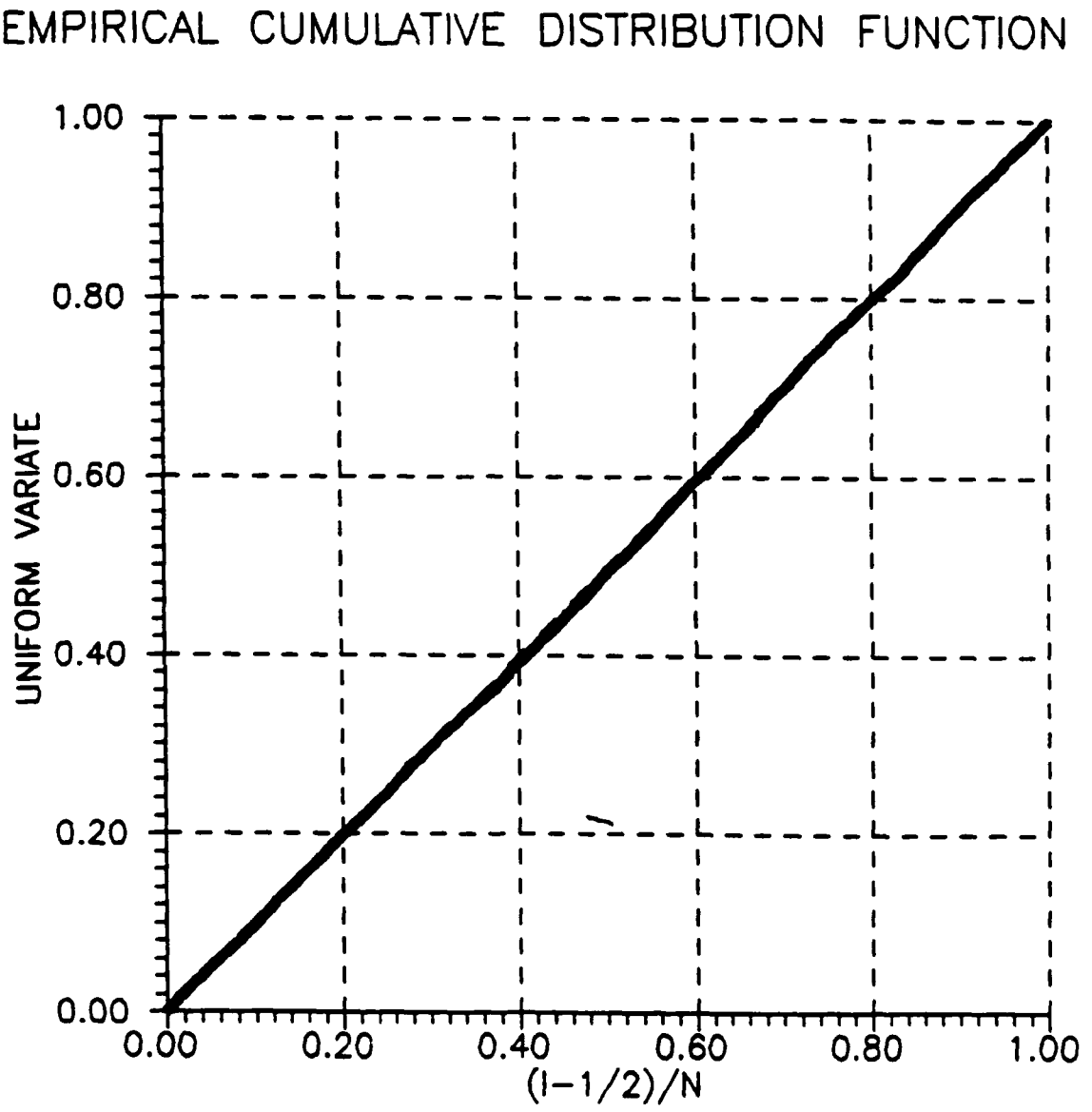

Figure 4.1: Empirical Cumulative Distribution Function, Original Algorithm

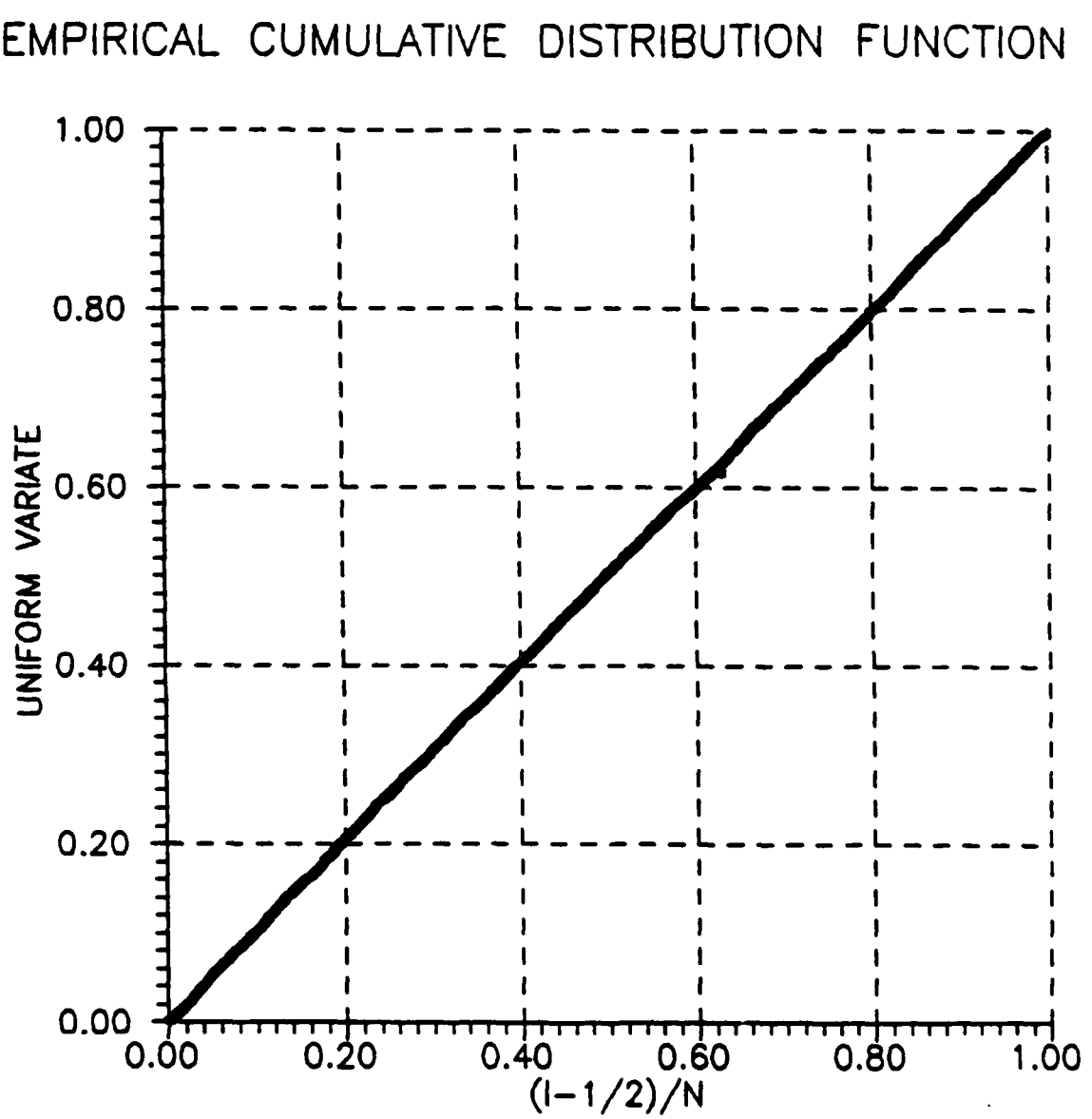

Figure 4.2: Empirical Cumulative Distribution Function, New Algorithm

be generated if desired. In addition, the algorithm for the uniform random number generator module has been included by Sky Computers in the microcode library of the VORTEX/AT in order to maximize the throughput gains possible when the VORTEX/AT is used. Fig. 4.3 is a plot of the empirical cumulative distribution function of 10,470 random numbers generated when the VORTEX/AT is used.

4.1.2 Gaussian Random Number Generator

The Gaussian variates used in the WCS are all derived from uniform variates using the Box–Muller sine–cosine technique [17]. This algorithm generates a pair of independent Gaussian random numbers (X_1, X_2) from a pair of indepedent uniform random numbers (U_1, U_2) on the interval $(0, 1)$ using the equations

$$X_1 = (-2 \ln U_1)^{1/2} \cos 2\pi U_2$$

$$X_2 = (-2 \ln U_1)^{1/2} \sin 2\pi U_2$$

The random numbers (X_1, X_2) will be from the same distribution with zero mean and unit variance.

The Gaussian random number generator module used in the baseline WCS, GRNG1R.FAP, produces only two Gaussian variates each time it is called. The Gaussian random number generator used in the enhanced WCS, like the uniform random number generator, produces all of the Gaussian variates needed for a process each time it is called. The algorithm for the Gaussian random number generator has also been included by Sky Computers in the microcode library of the VORTEX/AT

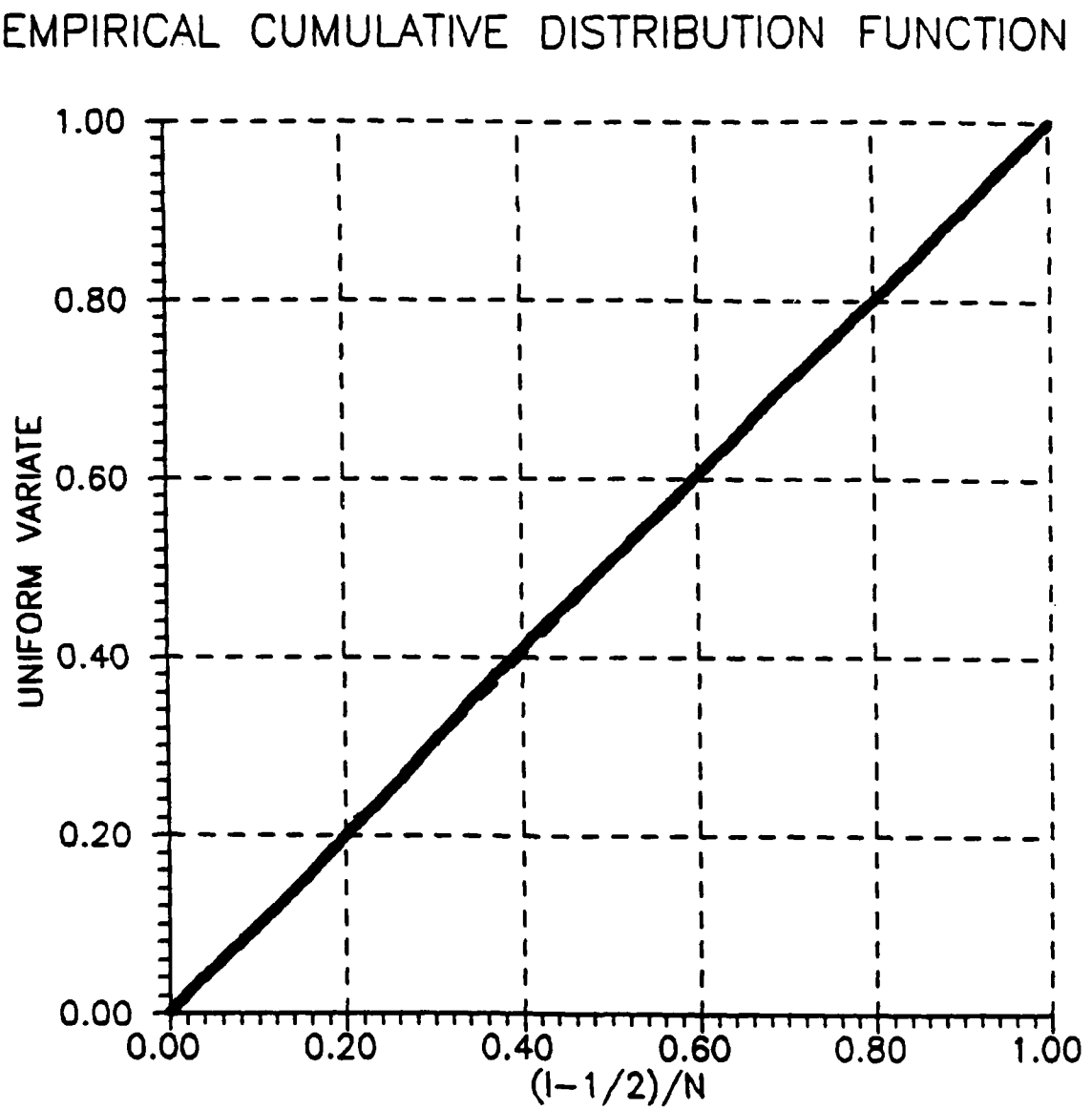

Figure 4.3: Empirical Cumulative Distribution Function, VORTEX/AT

to increase the throughput.

During the course of verifying the distributions of the Gaussian variates produced by both of the Gaussian random number generator modules, it was discovered that the VORTEX/AT microcoded version of the algorithm was implemented incorrectly. The random numbers produced by the microcoded random number generator did not exhibit the statistics of a standard normal distribution. In addition, the the same numbers were produced each time the random number generator module was called.

In an effort to implement a VORTEX/AT version of the Gaussian random number generator module for the generation of qualitative timing estimates, a new Gaussian random number generator module was developed. This new module implemented the Box–Muller sine–cosine technique entirely within VORTEX/AT, using DCI function calls. The uniform random numbers used in this module were generated using the VORTEX/AT microcoded uniform random number generator.

The statistics of the distributions of the random numbers produced by each of the three random number generator modules for a sequence of 10,470 numbers are tabulated in Table 4.1. The empirical cumulative distribution function is plotted in Figs. 4.4, 4.5, and 4.6 for each of these sequences.

The percentile probability plot (P–P plot) [16] for each of the distributions is plotted in Figs. 4.7, 4.8, and 4.9. The P–P plot is a useful graphical tool for comparing an unknown distribution with a known distribution, emphasizing the differences

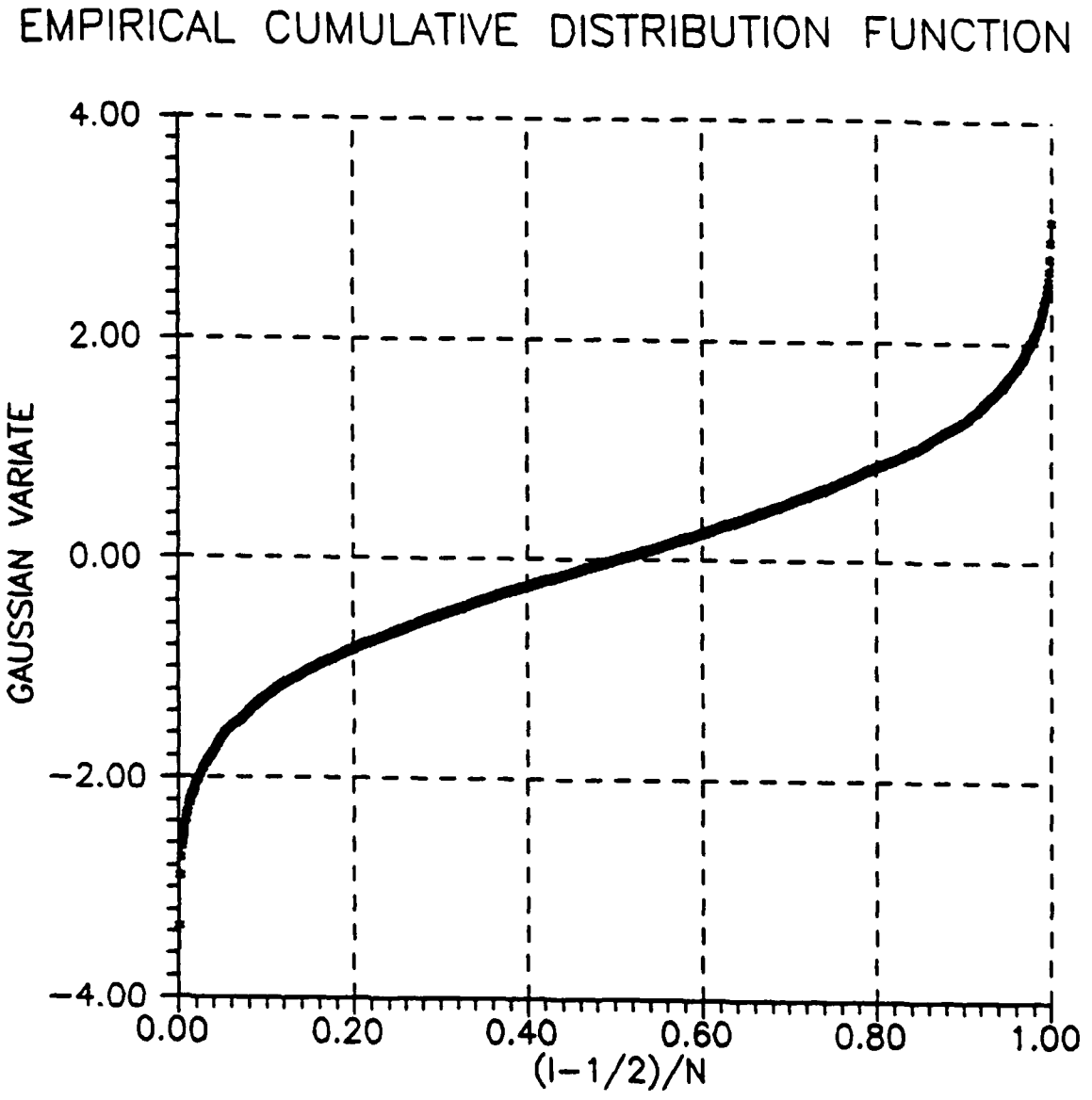

Figure 4.4: Empirical Cumulative Distribution Function, Baseline WCS Gaussian Random Number Generator

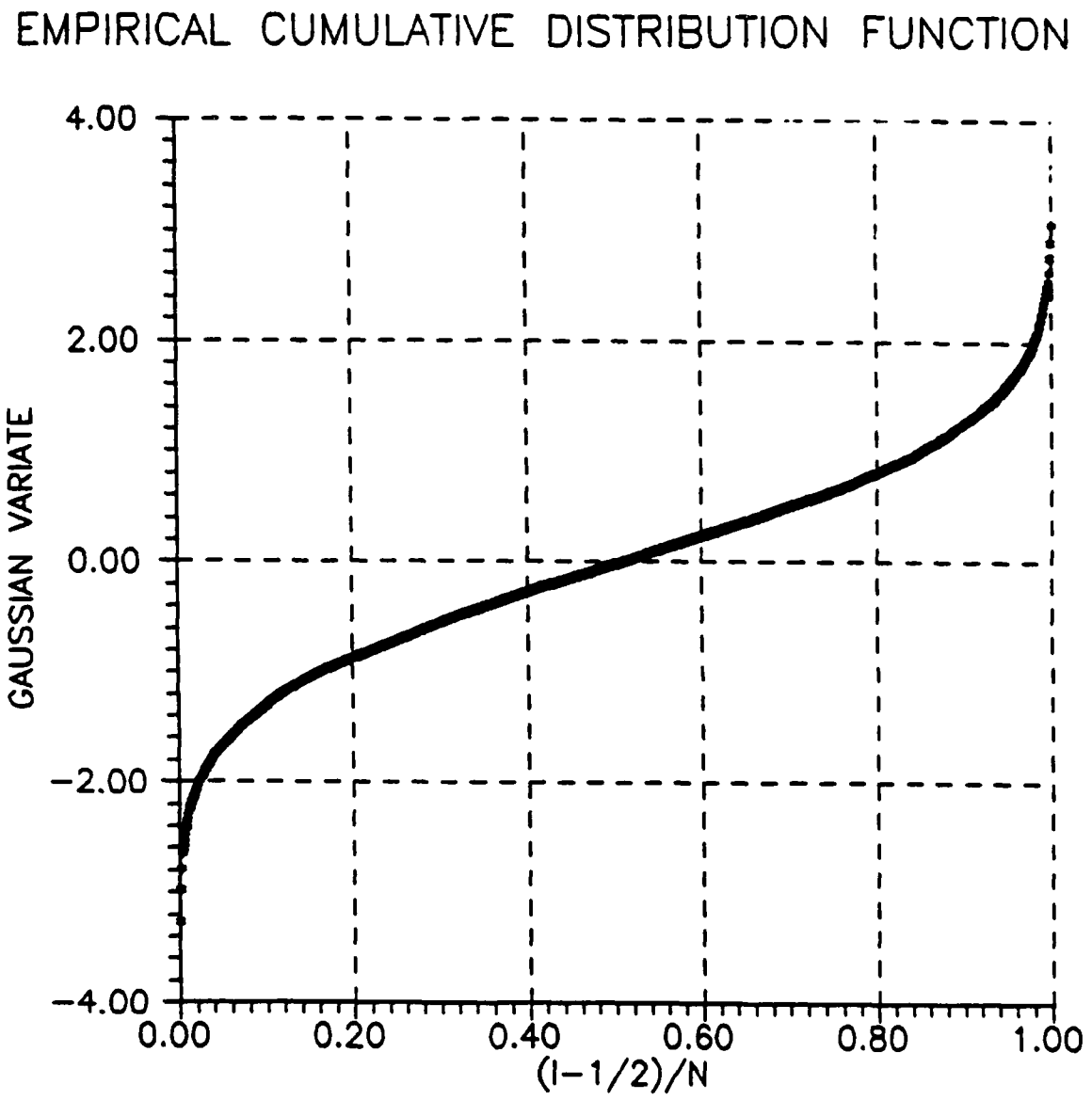

Figure 4.5: Empirical Cumulative Distribution Function, Enhanced WCS Gaussian Random Number Generator

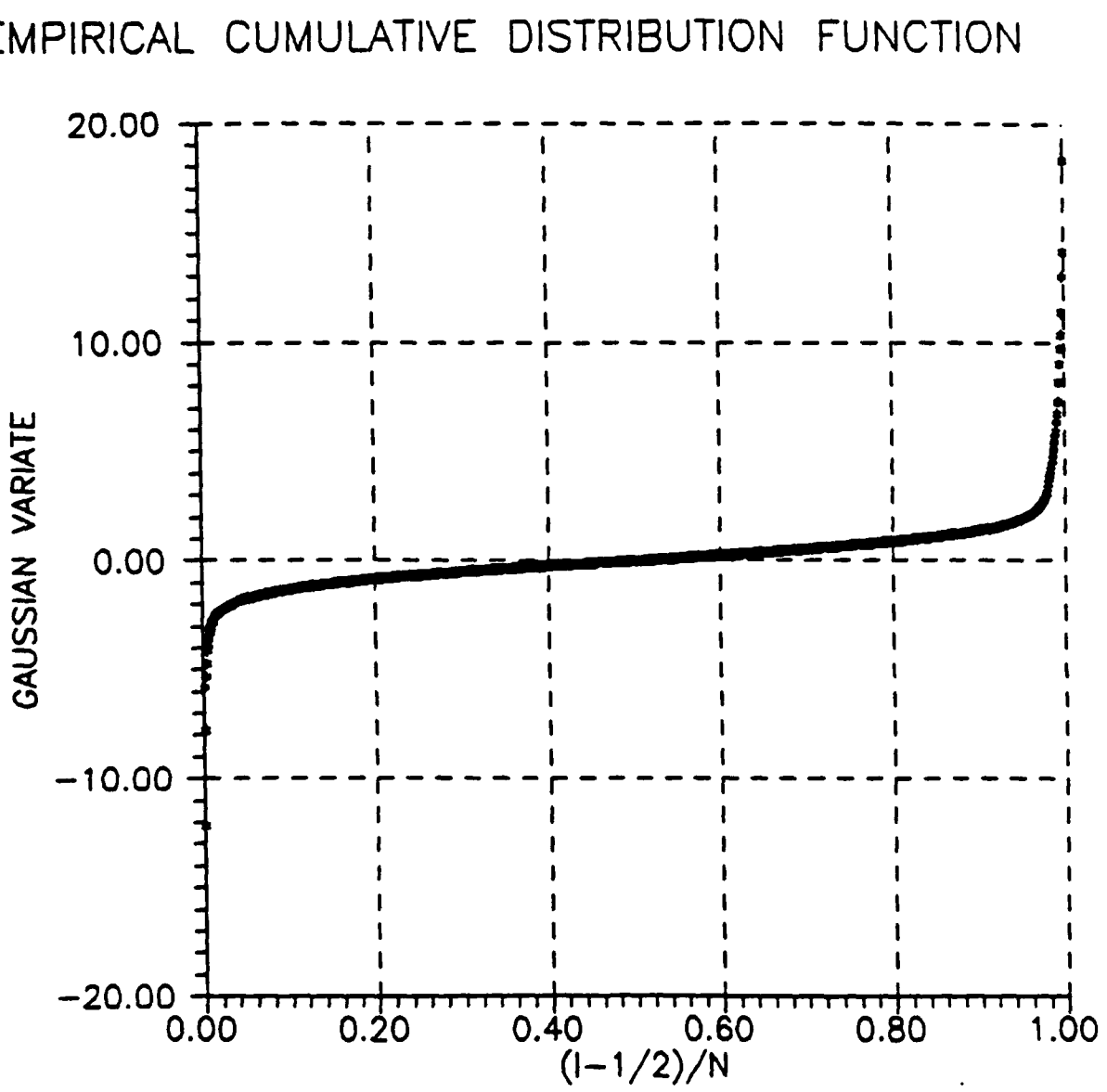

Figure 4.6: Empirical Cumulative Distribution Function, VORTEX/AT Microcode Gaussian Random Number Generator

Statistics for Generated Distributions			
	Baseline WCS Gaussian	Enhanced WCS Gaussian	VORTEX/AT Microcode
Mean	$+1.29 \times 10^{-2}$	-2.16×10^{-2}	$+1.10 \times 10^{-1}$
Median	-1.29×10^{-4}	-2.13×10^{-2}	-1.73×10^{-2}
Variance	0.989	0.986	2.797
Standard Deviation	0.995	0.993	1.673

Table 4.1: Statistics For Generated Gaussian Distributions

in the center of the distributions. If the distributions are identical, the plotted points will all lie along a straight line with unit slope. In Figs. 4.7, 4.8, and 4.9, the distributions of the generated random numbers are compared with the standard normal distribution. From these plots, it is evident that the random numbers generated by all three Gaussian random number generator modules exhibit the characteristics of the standard normal distribution in the center of the distribution.

The quantile probability plot (Q–Q plot) [16] for each of the distributions is plotted in Figs. 4.10, 4.11, and 4.12. The Q–Q plot, like the P–P plot, is useful in graphically comparing an unknown distribution with a known distribution. The Q–Q plot, however, emphasizes the differences in the tail regions. From the plots, it is evident that while the distributions of the random numbers produced by the orginal and the non–microcoded versions of the Gaussian random number generator modules exhibit the characteristics of the standard normal distribution everywhere,

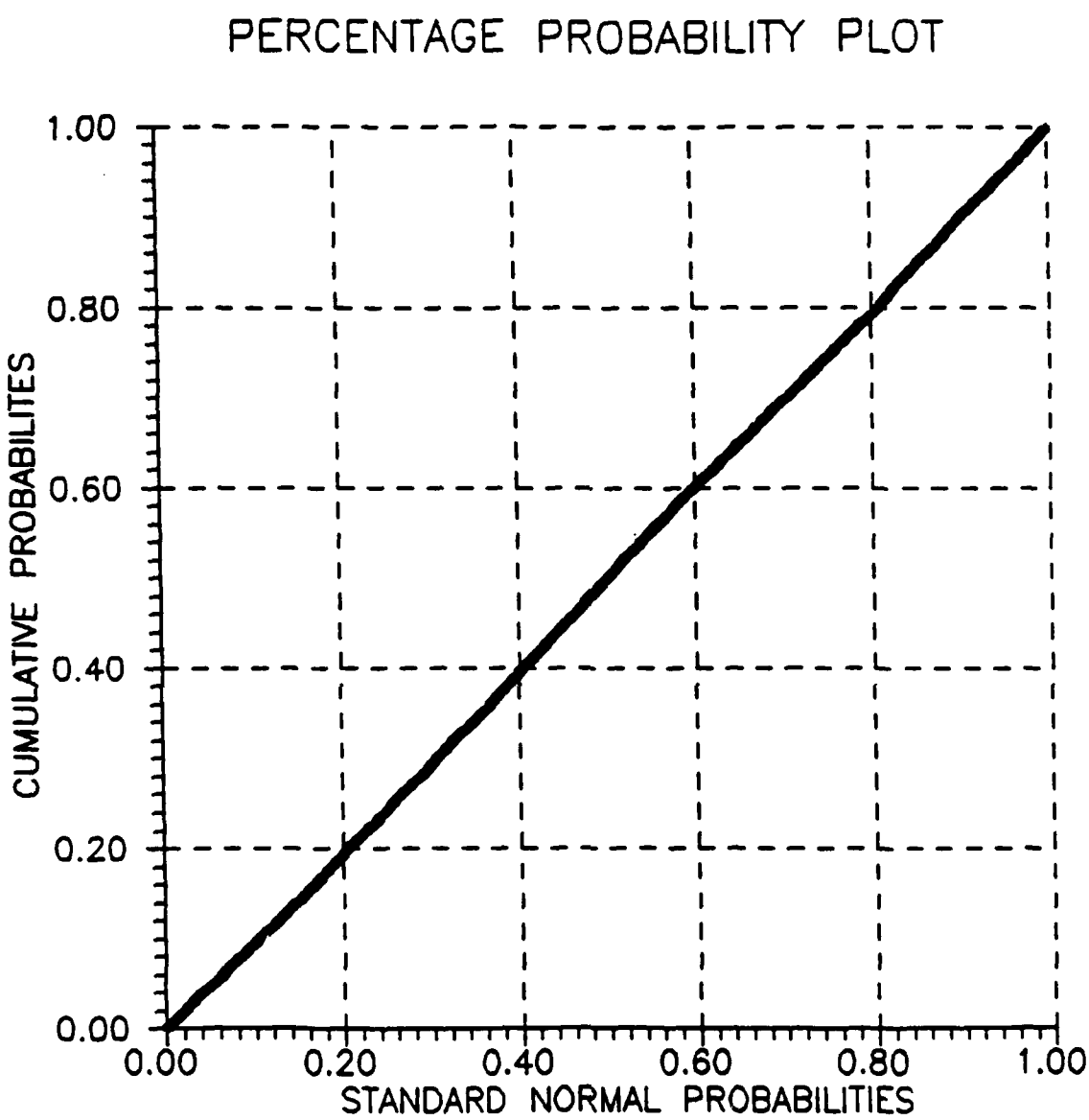

Figure 4.7: P–P Plot, Baseline WCS Gaussian Random Number Generator

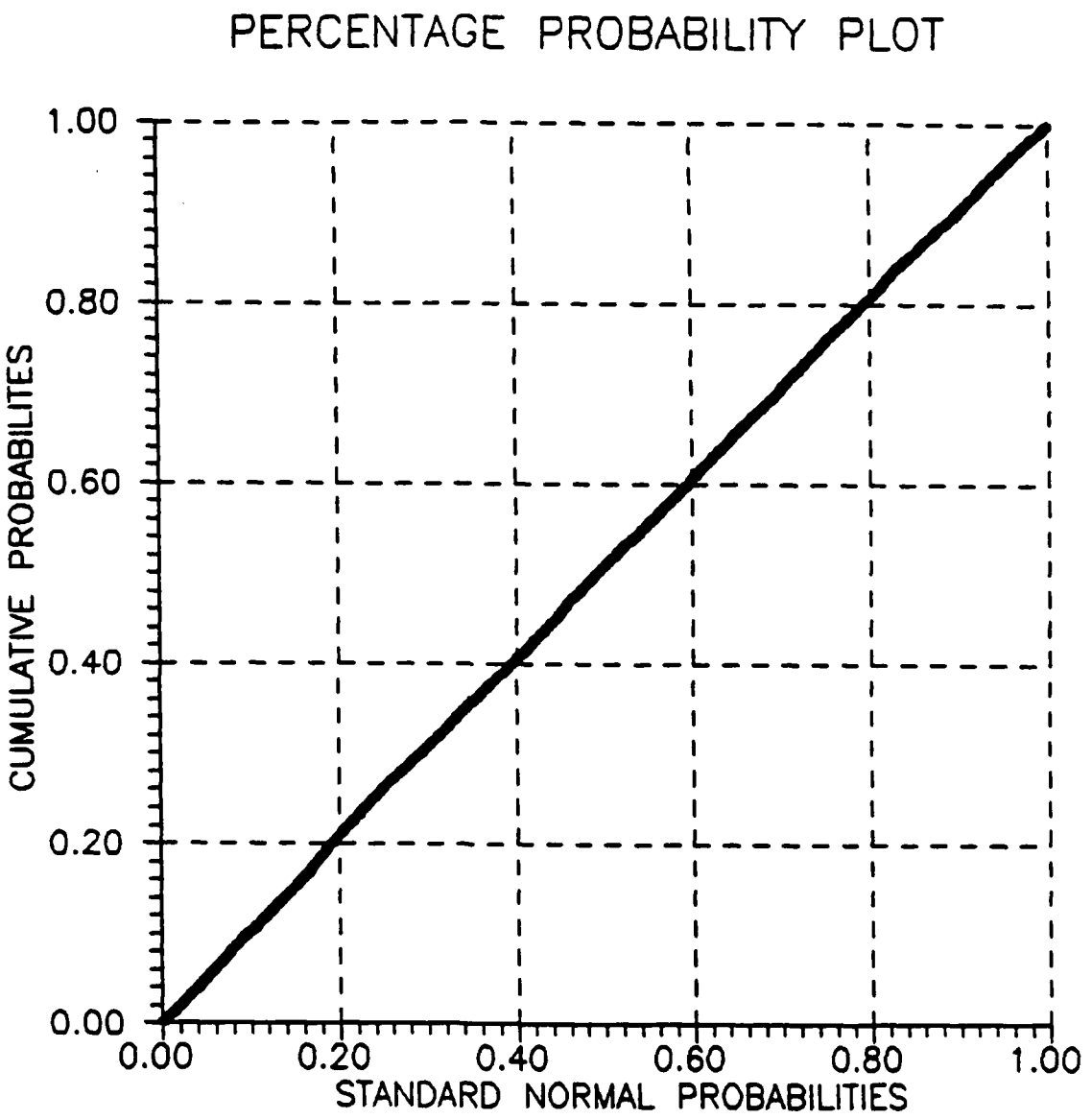

Figure 4.8: P–P Plot, Enhanced WCS Gaussian Random Number Generator

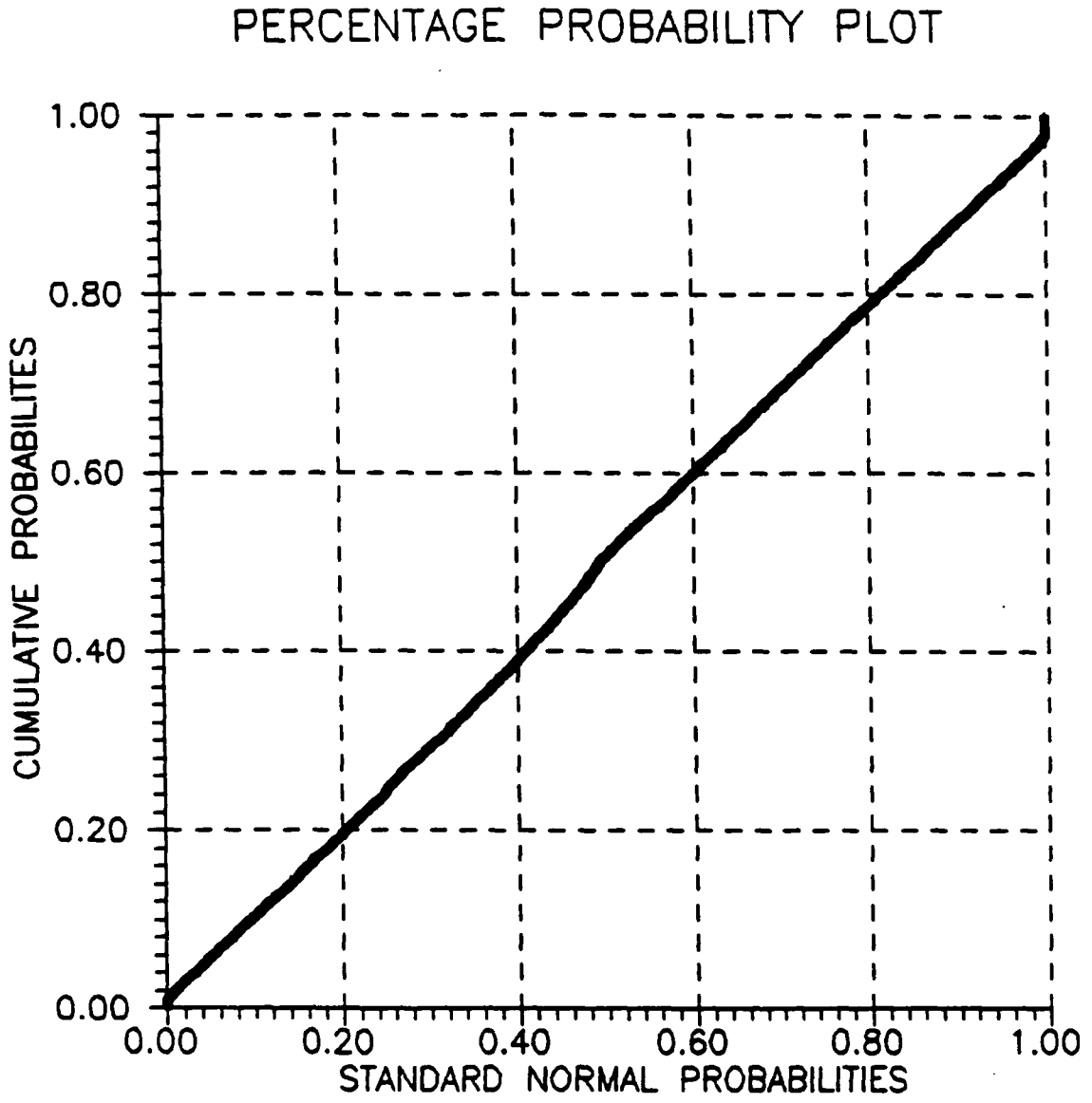

Figure 4.9: P–P Plot, VORTEX/AT Microcode Gaussian Random Number Generator

the distribution of the microcoded version does only in the center of the distribution. Figure 4.13 is a Q–Q plot of the distribution of the random numbers produced by the VORTEX/AT microcoded Gaussian random number generator compared with a Gaussian distribution having a standard deviation of 1.67, the same as that calculated for the generated distribution. This plot again exhibits differences in the tail regions, but in addition, also exhibits differences in the center of the distribution.

4.2 BINARY SOURCE MODULES

The binary source modules used in the enhanced WCS all have the capability of using the VORTEX/AT array processor if it is available. When a VORTEX/AT is not available, the modules operate in the same manner as in the baseline WCS.

4.2.1 Equiprobable Binary Source

The equiprobable binary source module, ESRC.FAP, generates a pseudo-random binary input sequence for the WCS. This binary sequence is derived from uniform variates (U_i) generated by the uniform random number generator modules. The uniform variates are mapped into the binary digits 0 and 1 through the mapping

$$x = \begin{cases} 0; & \text{if } U_i \leq 0.5 \\ 1; & \text{if } U_i > 0.5 \end{cases}$$

As a simple test of the equiprobable binary source module, a sequence of 10,000 binary digits was generated. This sequence contained 4998 binary 0's and 5002

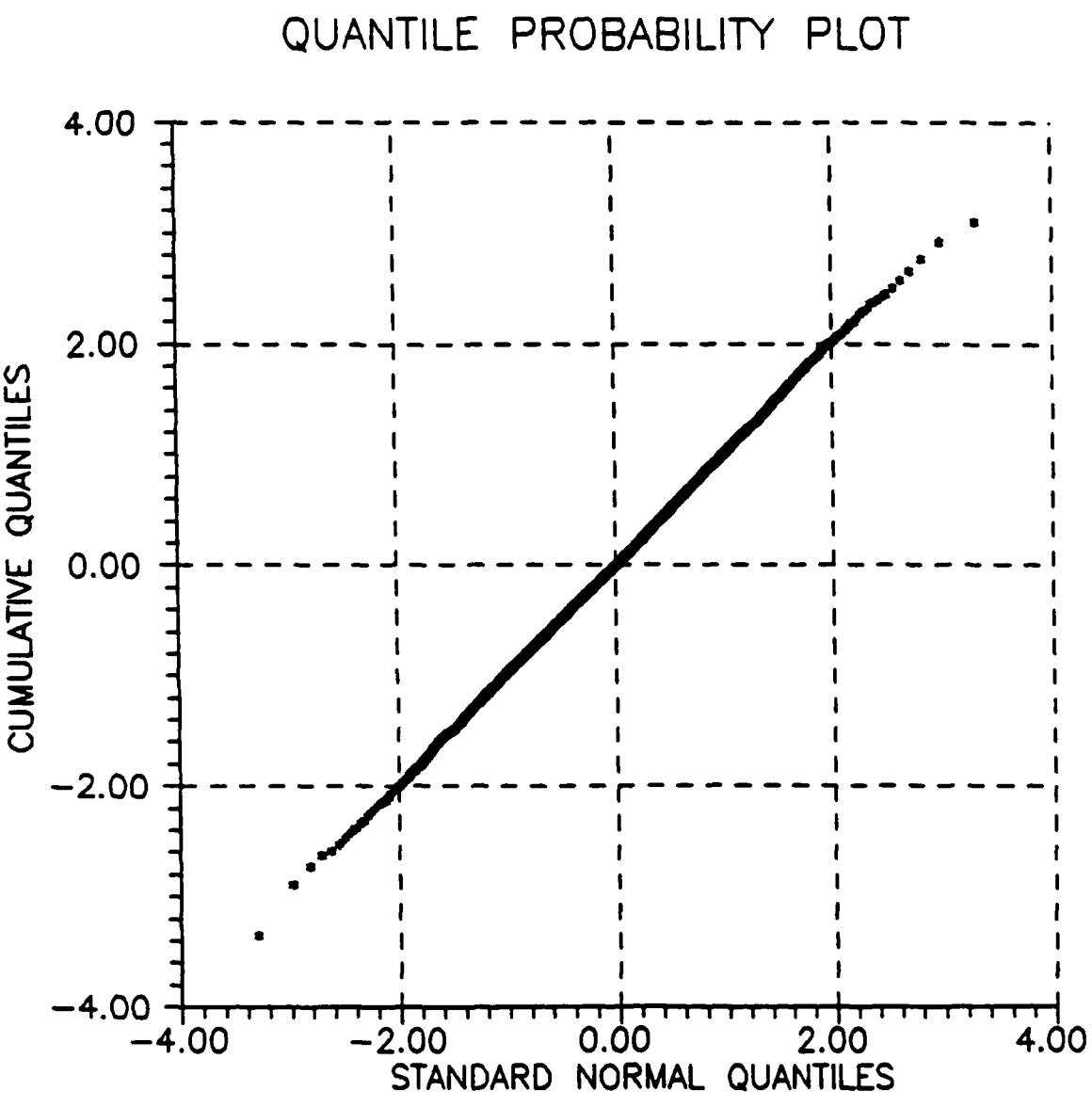

Figure 4.10: Q–Q Plot, Baseline WCS Gaussian Random Number Generator

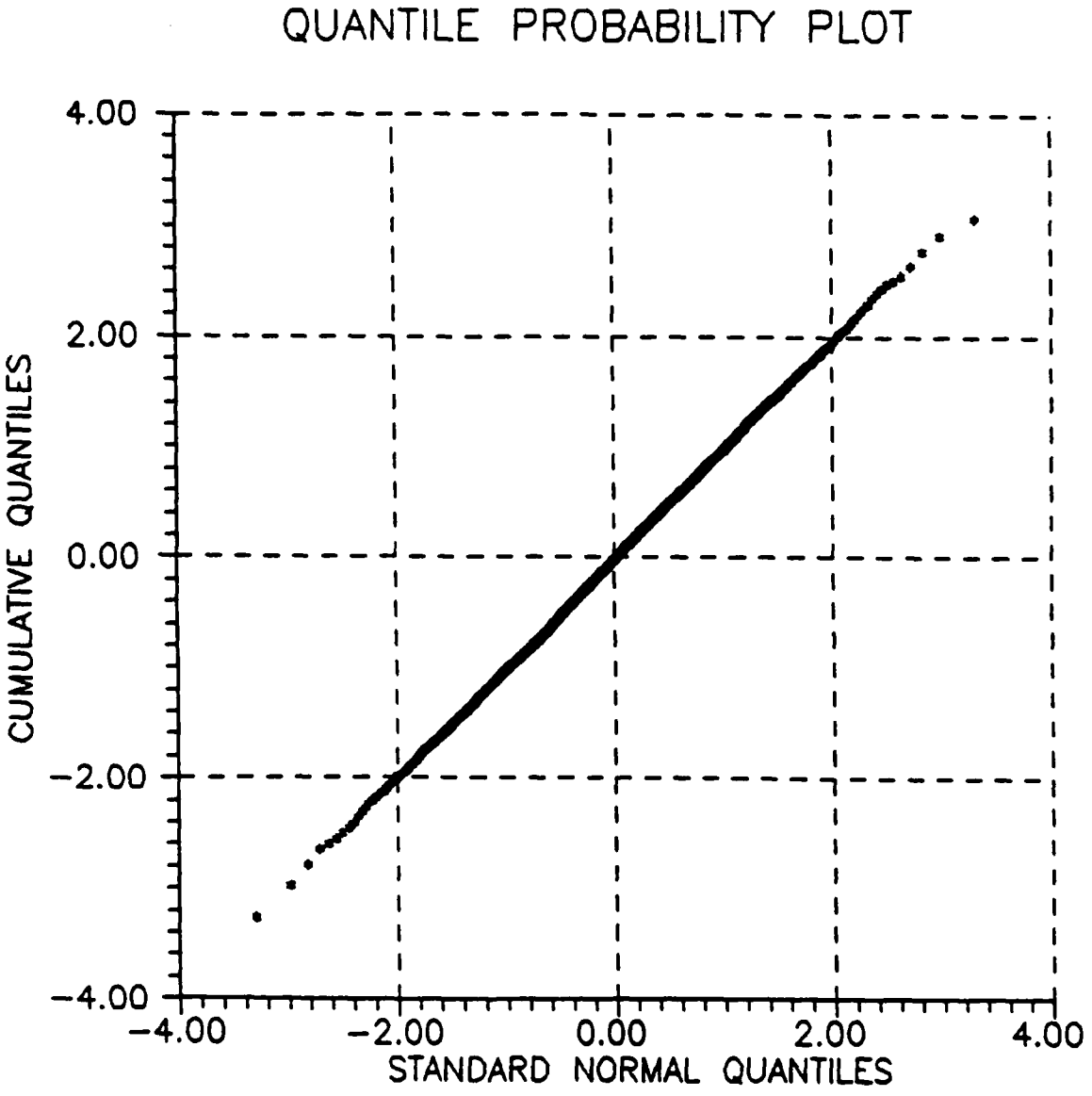

Figure 4.11: Q–Q Plot, Enhanced WCS Gaussian Random Number Generator

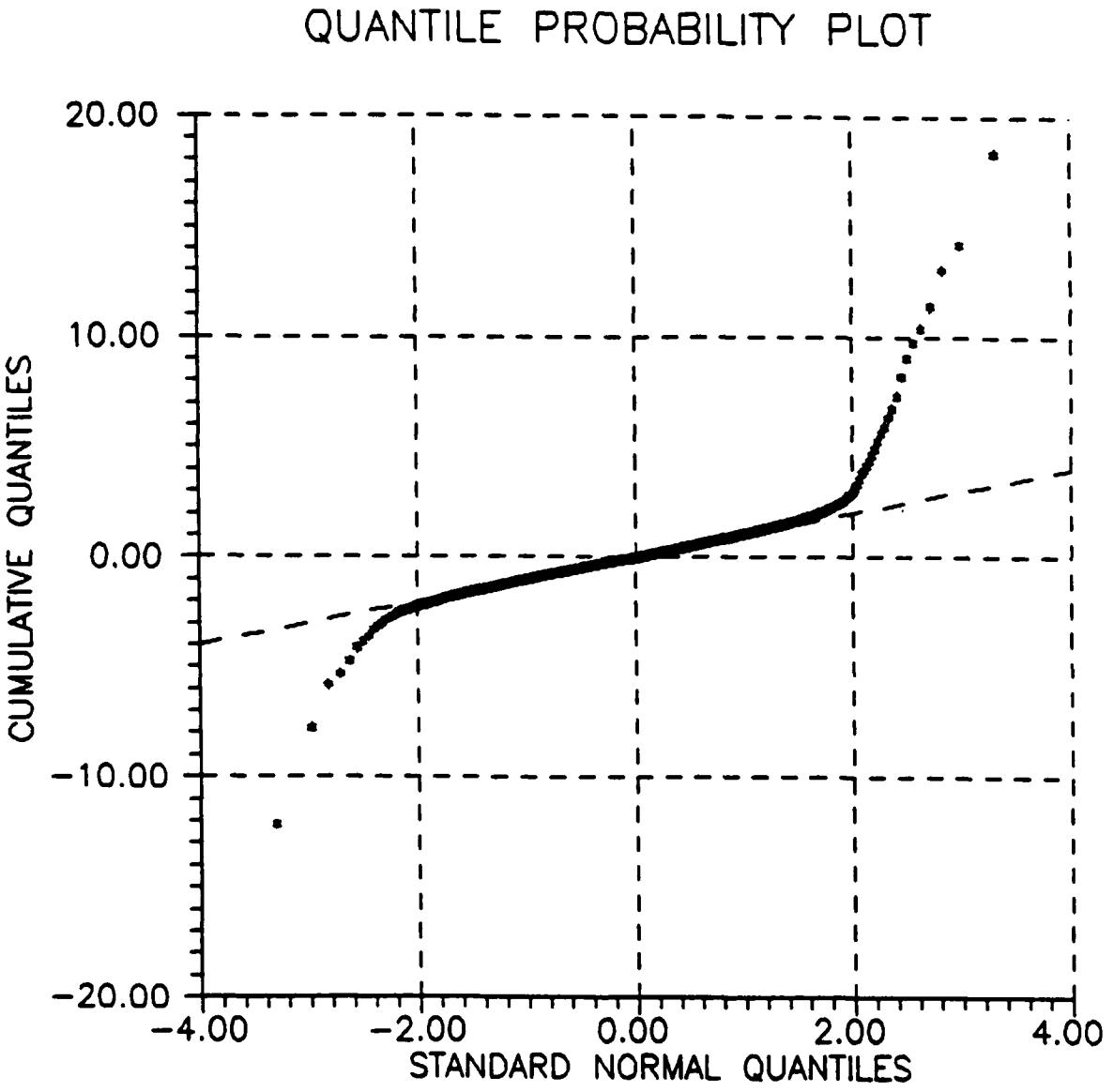

Figure 4.12: Q–Q Plot, VORTEX/AT Microcode Gaussian Random Number Generator

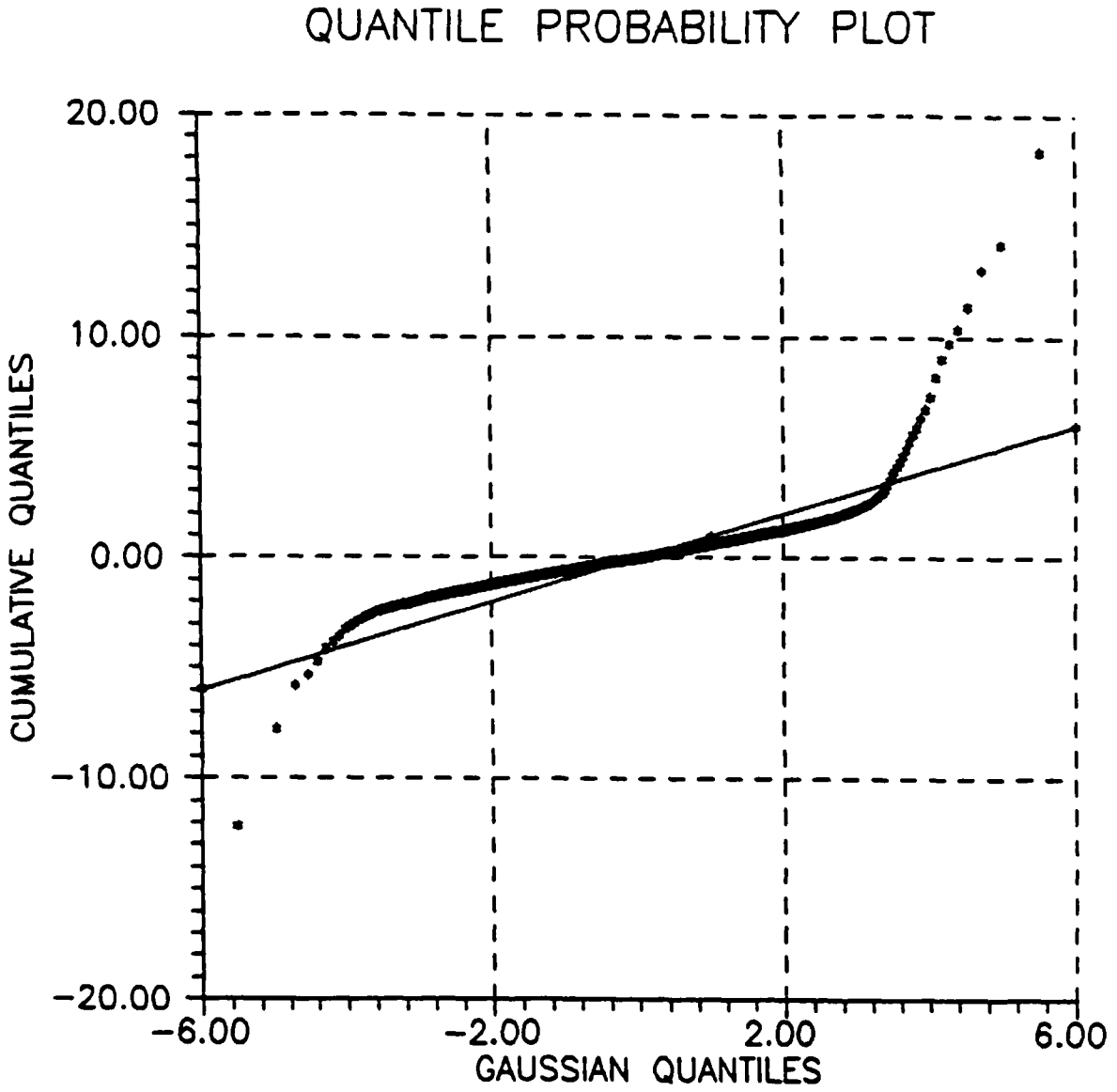

Figure 4.13: Q–Q Plot, VORTEX/AT Microcode Gaussian Random Number Generator vs Gaussian with 1.67 Standard Deviation

binary 1's when the VORTEX/AT was not used and 4959 binary 0's and 5041 binary 1's when the VORTEX/AT was used.

4.2.2 All Ones Binary Source

The all ones binary source module, OSRC.FAP, was included in the enhanced WCS as a diagnostic tool to aid in the development of new modules. This module simply generates a binary input sequence that consists entirely of binary 1's.

4.2.3 All Zeros Binary Source

The all zeros binary source module, ZSRC.FAP, is identical to the all ones binary source module except that it generates a binary input sequence that consists entirely of binary 0's.

4.3 MODULATOR/TRANSMITTER MODULES

The modulator/transmitter modules in the enhanced WCS have all been rewritten to allow for the future use of the VORTEX/AT in their implementation. While the original intention was to rewrite these modules to make full use of the VORTEX/AT if it was available, this proved to be an impossible task without the aid of the VEX preprocessor. The current versions of the enhanced WCS modulator/transmitter modules do not utilize the VORTEX/AT array processor.

4.3.1 BPSK Modulator/Transmitter

The binary phase shift keying (BPSK) modulator/transmitter module, XBPS.FAP, simulates the operation of a BPSK modulator/transmitter. When BPSK modulation is used, the phase of the RF carrier is shifted by $+\frac{\pi}{2}$ radians or $-\frac{\pi}{2}$ radians, depending on whether the data bit is a binary 0 or a binary 1. Binary phase shift keying can be viewed as binary phase modulation or binary amplitude modulation; the only requirement is that, during each signaling interval, $s(t)$ is one of two sinusoidal signals that differ by π radians.

In the enhanced WCS, the explicit details of the RF portions of a communication system are not modeled. Therefore, only the complex envelope representation of the channel signaling waveform is required. The complex envelope $\tilde{s}(t)$ can be represented as

$$\tilde{s}(t) = s_c(t) + js_s(t)$$

where $s_c(t)$ and $s_s(t)$ are the inphase and quadrature components, respectively. For the current case of BPSK modulation

$$s_c(t) = x_i\sqrt{\frac{2E_s}{T_s}}\cos\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.1}$$

$$s_s(t) = x_i\sqrt{\frac{2E_s}{T_s}}\sin\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.2}$$

where $x_i = \pm 1$ depending upon whether a binary 0 or a binary 1 is to be communicated. The quantity E_s in equations 4.1 and 4.2 represents the channel signal energy

per baud and the quantity T_s represents the channel symbol interval duration. The quantity θ_0 represents the phase uncertainty at the receiver.

4.3.2 DPSK Modulator/Transmitter

The differential phase shift keying (DPSK) modulator/transmitter module, XDPS.FAP, simulates the operation of a DPSK modulator/transmitter [18]. Coherent DPSK modulation is similar to BPSK modulation except that the output waveform is dependent upon the differential encoding of the current input data bit and the previous input data bit. The net effect of this process is to produce an output which changes only when successive input bits are different.

The inphase and quadrature components of the complex envelope for DPSK modulation can be represented as

$$s_c(t) = y_i\sqrt{\frac{2E_s}{T_s}}\cos\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.3}$$

$$s_s(t) = y_i\sqrt{\frac{2E_s}{T_s}}\sin\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.4}$$

where $y_i = \pm 1$ is the result of differentially encoding the current data bit and the previous data bit. The sequence $\{y_i\}$ is generated recursively according to

$$y_i = x_i y_{i-1}$$

where $x_i = \pm 1$ depending upon whether a binary 0 or a binary 1 is to be communicated and y_0 is arbitrarily taken to be $y_0 = -1$. As in the case of BPSK, the

quantity E_s in equations 4.3 and 4.4 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity θ_0 represents the phase uncertainty at the receiver.

4.3.3 QPSK Modulator/Transmitter

The quadrature phase shift keying (QPSK) modulator/transmitter module, XQPS.FAP, simulates the operation of a QPSK modulator/transmitter. The QPSK signal is of the same basic form as BPSK except that quaternary modulation is employed instead of binary modulation. Quadrature phase shift keying can be viewed as quadrature phase modulation or quadrature amplitude modulation; the only requirement is that, during each signaling interval, $s(t)$ is one of four sinusoidal signals that differ by $\frac{\pi}{2}$ radians.

In the enhanced WCS, QPSK modulation is implemented by using alternate bits of the input data to generate the inphase and quadrature components of $s(t)$. This can be represented as

$$s_c(t) = x_o\sqrt{\frac{2E_s}{T_s}}\cos\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.5}$$

$$s_s(t) = x_e\sqrt{\frac{2E_s}{T_s}}\sin\theta_0; \qquad (i-1)T_s \leq t < iT_s \tag{4.6}$$

where $x_o = \pm 1$ is the sequence of odd input data bits and $x_e = \pm 1$ is the sequence of even input data bits to be communicated. The quantity E_s in equations 4.5 and 4.6 represents the channel signal energy per baud and the quantity T_s represents the

channel symbol interval duration. The quantity θ_0 represents the phase uncertainty at the receiver.

For most purposes, it is better to compare BPSK and QPSK signals that have the same energy per data bit and the same data rate. If BPSK and QPSK are compared for the same data rate, BPSK requires twice the bandwidth of QPSK. If the two signals have the same data rate and equal power, they also have the same energy per data bit.

4.3.4 OQPSK Modulator/Transmitter

The offset quadrature phase shift keying (OQPSK) modulator/transmitter module, XOQP.FAP, simulates the operation of a OQPSK modulator/transmitter. The OQPSK signal is of the same basic form as QPSK except that the baseband signal of the inphase component is offset by $T_s/2$ relative to the baseband signal of the quadrature component. This offset leads to a more nearly constant envelope of the modulated waveform than in the case of BPSK or QPSK since both the inphase and quadrature components can not transition at the same time. This can be represented as

$$s_c(t) = x_o\sqrt{\frac{2E_s}{T_s}}\cos\theta_0; \qquad (i-1)T_s \le t < iT_s \tag{4.7}$$

$$s_s(t) = x_e\sqrt{\frac{2E_s}{T_s}}\sin\theta_0; \qquad (i-\frac{1}{2})T_s \le t < (i+\frac{1}{2})T_s \tag{4.8}$$

where $x_o = \pm 1$ is the sequence of odd input data bits and $x_e = \pm 1$ is the sequence of even input data bits to be communicated. The quantity E_s in equations 4.7 and 4.8 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity θ_0 represents the phase uncertainty at the receiver.

In the enhanced WCS, OQPSK is implemented by delaying the transitions in the quadrature component by one half of the channel symbol time T_s. The last value of the quadrature component processed is saved to be used the next time the module is called.

4.4 CHANNEL MODULES

The channel modules used in the enhanced WCS have the capability to use the VORTEX/AT array processor if it is available. When a VORTEX/AT is not available, the modules operate in the same manner as in the baseline WCS.

4.4.1 AWGN Channel

The additive white Gaussian noise (AWGN) channel module, AWGN.FAP, simulates the effects of additive white Gaussian noise on the transmitted signal. The net effect of the AWGN channel module is to introduce an additive noise component $n(t)$ to the output of the modulator/transmitter module $s(t)$ to produce the channel

output $r(t)$. This relationship is given by

$$r(t) = s(t) + n(t).$$

The complex envelopes of $r(t)$ and $s(t)$ can be expressed in terms of their I/Q components according to the equations

$$\tilde{r}(t) = r_c(t) + jr_s(t)$$

$$\tilde{s}(t) = s_c(t) + js_s(t).$$

The complex envelope of $n(t)$ can be expressed in terms of its I/Q components according to the equation

$$\tilde{n}(t) = n_c(t) + jn_s(t).$$

where $n_c(t)$ and $n_s(t)$ are mutually independent real white Gaussian noise processes with the common autocorrelation function

$$R_{n_c n_c}(\tau) = R_{n_s n_s}(\tau) = \frac{N_0}{2}\delta(\tau).$$

Thus, the channel output can be expressed in terms of its inphase and quadrature components as

$$r_c(t) = s_c(t) + n_c(t)$$

$$r_s(t) = s_s(t) + n_s(t).$$

Implementing this digitally results in the following equations

$$r_{c_i} = s_{c_i} + n_{c_i}$$

$$r_{s_i} = s_{s_i} + n_{s_i}$$

where n_{c_i} and n_{s_i} are mutually independent identically distributed Gaussian variates with zero mean and variance $N_0/2$.

To obtain simulation results for various values of the ratios E_s/N_0 or E_b/N_0, either the magnitude of the signal component or of the noise component or of both could be varied. It has been proven convenient[19] to adjust the magnitude of the signal component while holding the magnitude of the noise component constant. This allows the noise component to be taken from a Gaussian sequence with zero mean and unit variance.

The AWGN channel module in the enhanced WCS performs this simulation by simply adding Gaussian variates to each element of the modulator/transmitter output sequence. The signal components are scaled in the modulator/transmitter module prior to this addition.

4.4.2 Inverting Channel

The inverting channel module, INVRT.FAP, was included in the enhanced WCS as a diagnostic tool to aid in the development of new modules. This module inverts the modulator/transmitter output sequence by changing each binary 0 to a binary 1 and each binary 1 to a binary 0. This inversion is useful in verifying that errors occur when they should in a simulation.

4.5 DEMODULATOR/RECEIVER MODULES

The demodulator/receiver modules in the enhanced WCS have all been rewritten to allow for the future use of the VORTEX/AT in their implementation. While the original intention was to rewrite these modules to make full use of the VORTEX/AT if it was available, this proved to be an impossible task without the aid of the VEX preprocessor. The current versions of the enhanced WCS demodulator/receiver modules do not utilize the VORTEX/AT array processor.

4.5.1 BPSK Demodulator/Receiver

The binary phase shift keying demodulator/receiver module, RBPS.FAP, simulates the operation of a BPSK demodulator/receiver. The matched filter receiver operates by calculating the statistic

$$r_i = \int_{(i-1)T_s}^{iT_s} [r_c(t)\hat{s}_c(t) + r_s(t)\hat{s}_s(t)]\, dt; \qquad i = 1, 2, \ldots \tag{4.9}$$

during each successive baud interval. The quantities $r_c(t)$ and $r_s(t)$ are the inphase and quadrature components of the received signal and the quantities $\hat{s}_c(t)$ and $\hat{s}_s(t)$ are local estimates of the inphase and quadrature signal components. In particular,

$$\hat{s}_c(t) = \sqrt{\frac{2E_s}{T_s}} \cos\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.10}$$

$$\hat{s}_s(t) = \sqrt{\frac{2E_s}{T_s}} \sin\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.11}$$

for $i = 1, 2, \ldots$. The quantity E_s in equations 4.10 and 4.11 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity $\hat{\theta}_i$ represents the local phase estimate computed during the ith baud interval.

In the enhanced WCS, since this operation is implemented digitally, the discrete version of equation 4.9 is defined by

$$r'_i = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) + r'_s(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) \right]; \qquad i = 1, 2, \ldots$$

The quantities $r'_c(k\frac{T_s}{N})$ and $r'_s(k\frac{T_s}{N})$, $k = 0, 1, \ldots, N-1$ are normalized versions of the inphase and quadrature components of the channel output during the ith baud interval. The quantity N is the number of samples per baud. The quantities $\hat{s}'_c(k\frac{T_s}{N})$ and $\hat{s}'_s(k\frac{T_s}{N})$ are defined by

$$\hat{s}'_c(k\frac{T_s}{N}) = \cos\hat{\theta}_i \qquad k = 1, 2, \ldots, N-1$$

$$\hat{s}'_s(k\frac{T_s}{N}) = \sin\hat{\theta}_i. \qquad k = 1, 2, \ldots, N-1$$

The sequence $\{r'_i\}$ is then passed to the decoder which implements a hard-decision rule according to

$$\hat{x}_i = \begin{cases} -1; & \text{if } r'_i \leq 0 \\ +1; & \text{if } r'_i > 0 \end{cases}$$

4.5.2 DPSK Demodulator/Receiver

The differential phase shift keying demodulator/receiver module, DPBS.FAP, simulates the operation of a DPSK demodulator/receiver [18]. The matched filter

receiver operates by calculating the statistic

$$z_i = \int_{iT_s}^{(i+1)T_s} \left[\tilde{r}(t)\hat{\tilde{s}}^*(t)\right]\, dt; \qquad i = 0, 1, \ldots$$

during each successive baud interval. The quantity $\tilde{r}(t)$ is the complex envelope of the received signal and the quantity $\hat{\tilde{s}}(t)$ is the complex envelope of the local estimate of the signal. In particular,

$$\hat{s}_c(t) = \sqrt{\frac{2E_s}{T_s}} \cos\hat{\theta}_a; \qquad (i)T_s \leq t < (i+1)T_s \tag{4.12}$$

$$\hat{s}_s(t) = \sqrt{\frac{2E_s}{T_s}} \sin\hat{\theta}_a; \qquad (i)T_s \leq t < (i-1)T_s \tag{4.13}$$

for $i = 1, 2, \ldots$. The quantity E_s in equations 4.12 and 4.13 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity $\hat{\theta}_a$ represents a completely arbitrary phase reference which will be assumed to be constant throughout the entire message sequence.

In the enhanced WCS, since this operation is implemented digitally, the discrete version of equation 4.5.2 is defined by

$$z'_{c_i} = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) + r'_s(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N})\right]; \qquad i = 1, 2, \ldots$$

$$z'_{s_i} = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) - r'_s(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N})\right]; \qquad i = 1, 2, \ldots$$

The quantities $r'_c(k\frac{T_s}{N})$ and $r'_s(k\frac{T_s}{N})$, $k = 0, 1, \ldots, N-1$ are normalized versions of the inphase and quadrature components of the channel output during the ith baud

interval. The quantity N is the number of samples per baud. The quantities $\hat{s}'_c(k\frac{T_s}{N})$ and $\hat{s}'_s(k\frac{T_s}{N})$ are defined by

$$\hat{s}'_c(k\frac{T_s}{N}) = \cos\hat{\theta}_a \qquad k = 1, 2, \ldots, N-1$$
$$\hat{s}'_s(k\frac{T_s}{N}) = \sin\hat{\theta}_a. \qquad k = 1, 2, \ldots, N-1$$

The normalized receiver output sequence $\{r'_i\}$ is defined as

$$r'_i = \mathrm{Re}\{z_i z^*_{i-1}\}; \qquad i = 1, 2, \ldots$$

where, by convention, $r'_0 = \mathrm{Re}\{z_0\}$. The sequence $\{r'_i\}$ is then passed to the decoder which implements a hard-decision rule according to

$$\hat{x}_i = \begin{cases} -1; & \text{if } r'_i \leq 0 \\ +1; & \text{if } r'_i > 0 \end{cases}$$

4.5.3 QPSK Demodulator/Receiver

The quadrature phase shift keying demodulator/receiver module, QRQR.FAP, simulates the operation of a QPSK demodulator/receiver. The matched filter receiver operates by calculating the statistics

$$z_{i_c} = \int_{(i-1)T_s}^{iT_s} [r_c(t)\hat{s}_c(t) + r_s(t)\hat{s}_s(t)]\, dt; \qquad i = 1, 2, \ldots \tag{4.14}$$

$$z_{i_s} = \int_{(i-1)T_s}^{iT_s} [r_c(t)\hat{s}_s(t) - r_s(t)\hat{s}_c(t)]\, dt; \qquad i = 1, 2, \ldots \tag{4.15}$$

during each successive channel symbol interval. The quantities $r_c(t)$ and $r_s(t)$ are the inphase and quadrature components of the received signal and the quantities

$\hat{s}_c(t)$ and $\hat{s}_s(t)$ are local estimates of the inphase and quadrature signal components. In particular,

$$\hat{s}_c(t) = \sqrt{\frac{2E_s}{T_s}} \cos\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.16}$$

$$\hat{s}_s(t) = \sqrt{\frac{2E_s}{T_s}} \sin\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.17}$$

for $i = 1, 2, \ldots$. The quantity E_s in equations 4.16 and 4.17 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity $\hat{\theta}_i$ represents the local phase estimate computed during the ith channel symbol interval.

In the enhanced WCS, since this operation is implemented digitally, the discrete versions of equations 4.14 and 4.15 are defined by

$$z'_{i_o} = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) + r'_s(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) \right]; \qquad i = 1, 2, \ldots \tag{4.18}$$

$$z'_{i_e} = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) - r'_s(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) \right]; \qquad i = 1, 2, \ldots \tag{4.19}$$

The quantities $r'_c(k\frac{T_s}{N})$ and $r'_s(k\frac{T_s}{N})$, $k = 0, 1, \ldots, N-1$ are normalized versions of the inphase and quadrature components of the channel output during the ith baud interval. The quantity N is the number of samples per baud. The quantities $\hat{s}'_c(k\frac{T_s}{N})$ and $\hat{s}'_s(k\frac{T_s}{N})$ are defined by

$$\hat{s}'_c(k\frac{T_s}{N}) = \cos\hat{\theta}_i \qquad k = 1, 2, \ldots, N-1$$

$$\hat{s}'_s(k\frac{T_s}{N}) = \sin\hat{\theta}_i. \qquad k = 1, 2, \ldots, N-1$$

The sequences $\{z'_{i_o}\}$ and $\{z'_{i_e}\}$ are used to reconstruct the sequence $\{r'_i\}$ with $\{z'_{i_o}\}$ being the odd bits of $\{r'_i\}$ and $\{z'_{i_e}\}$ being the even bits of $\{r'_i\}$. The sequence $\{r'_i\}$ is then passed to the decoder which implements a hard-decision rule according to

$$\hat{x}_i = \begin{cases} -1; & \text{if } r'_i \leq 0 \\ +1; & \text{if } r'_i > 0 \end{cases}$$

4.5.4 OQPSK Demodulator/Receiver

The offset quadrature phase shift keying demodulator/receiver module, ROQP.FAP, simulates the operation of an OQPSK demodulator/receiver. The matched filter receiver operates by calculating the statistics

$$z_{i_o} = \int_{(i-1)T_s}^{iT_s} [r_c(t)\hat{s}_c(t) + r_s(t)\hat{s}_s(t)]\, dt; \qquad i = 1, 2, \ldots \tag{4.20}$$

$$z_{i_e} = \int_{(i-\frac{1}{2})T_s}^{(i+\frac{1}{2})T_s} [r_c(t)\hat{s}_s(t) - r_s(t)\hat{s}_c(t)]\, dt; \qquad i = 1, 2, \ldots \tag{4.21}$$

during each successive channel symbol interval. The quantities $r_c(t)$ and $r_s(t)$ are the inphase and quadrature components of the received signal and the quantities $\hat{s}_c(t)$ and $\hat{s}_s(t)$ are local estimates of the inphase and quadrature signal components. In particular,

$$\hat{s}_c(t) = \sqrt{\frac{2E_s}{T_s}} \cos\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.22}$$

$$\hat{s}_s(t) = \sqrt{\frac{2E_s}{T_s}} \sin\hat{\theta}_i; \qquad (i-1)T_s \leq t < iT_s \tag{4.23}$$

for $i = 1, 2, \ldots$. The quantity E_s in equations 4.22 and 4.23 represents the channel signal energy per baud and the quantity T_s represents the channel symbol interval duration. The quantity $\hat{\theta}_i$ represents the local phase estimate computed during the ith channel symbol interval.

In the enhanced WCS, since this operation is implemented digitally, the discrete versions of equation 4.20 and equation 4.21 are defined by

$$z'_{i_o} = \sum_{k=0}^{N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) + r'_s(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) \right]; \qquad i = 1, 2, \ldots \quad (4.24)$$

$$z'_{i_e} = \sum_{k=\frac{T_s}{2}}^{\frac{T_s}{2}+N-1} \left[r'_c(k\frac{T_s}{N})\hat{s}'_s(k\frac{T_s}{N}) + r'_s(k\frac{T_s}{N})\hat{s}'_c(k\frac{T_s}{N}) \right]; \qquad i = 1, 2, \ldots \quad (4.25)$$

The quantities $r'_c(k\frac{T_s}{N})$ and $r'_s(k\frac{T_s}{N})$, $k = 0, 1, \ldots, N-1$ are normalized versions of the inphase and quadrature components of the channel output during the ith baud interval. The quantity N is the number of samples per baud. The quantities $\hat{s}'_c(k\frac{T_s}{N})$ and $\hat{s}'_s(k\frac{T_s}{N})$ are defined by

$$\hat{s}'_c(k\frac{T_s}{N}) = \cos\hat{\theta}_i \qquad k = 1, 2, \ldots, N-1$$
$$\hat{s}'_s(k\frac{T_s}{N}) = \sin\hat{\theta}_i. \qquad k = 1, 2, \ldots, N-1$$

The sequences $\{z'_{i_o}\}$ and $\{z'_{i_e}\}$ are then used to reconstruct the sequence $\{r'_i\}$ with $\{z'_{i_o}\}$ being the odd bits of $\{r'_i\}$ and $\{z'_{i_e}\}$ being the even bits of $\{r'_i\}$. The sequence $\{r'_i\}$ is then passed to the decoder which implements a hard-decision rule according

to

$$\hat{x}_i = \begin{cases} -1; & \text{if } r'_i \leq 0 \\ +1; & \text{if } r'_i > 0 \end{cases}$$

4.6 BINARY SINK MODULES

The binary sink modules used in the enhanced WCS all have the capability of using the VORTEX/AT array processor if it is available. These modules all complement the corresponding binary source modules and have been tested in conjunction with the source modules to verify proper operation.

4.6.1 Equiprobable Binary Sink

The equiprobable binary sink module, BTCNTE.FAP, generates a pseudo-random binary sequence identical to the input binary sequence. This generated sequence is compared with the decoded received sequence on a bit-by-bit basis to determine the number of bit errors that have occurred in the simulated transmission. In addition, an output error sequence is generated which is the modulo-2 sum of the generated and received sequences. This output error sequence can be used to analyze the distribution of any errors that may have occurred.

4.6.2 All Ones Binary Sink

The all ones binary sink module, BTCNTO.FAP, compares each bit of the decoded received sequence with a binary 1 to determine the number of bit errors that

have occurred in the simulated transmission. In addition, like the equiprobable binary sink, an output error sequence is also generated for error distribution analysis purposes.

4.6.3 All Zeros Binary Sink

The all zeros binary sink module, BTCNTZ.FAP, is identical to the all ones binary sink except that each decoded received bit is compared to a binary 0.

4.7 VORTEX/AT SUPPORT MODULES

The VORTEX/AT support modules have been added to the enhanced WCS to allow for the optional use of the VORTEX/AT array processor. These modules are similar in function to modules used in the ICS to control the operation of the AP-120B array processor.

4.7.1 VORTEX/AT Initialize

The module APINIT.FAP is used to logically attach the VORTEX/AT to the enhanced WCS and initialize it prior to its being used. This module also transfers the common block APARY to the VORTEX/AT memory area where the data will be more readily available to the VORTEX/AT.

4.7.2 VORTEX/AT Release

The module APRLSE.FAP is used to relinquish control of the VORTEX/AT upon the completion of a program run.

4.8 MISCELLANEOUS SUPPORT MODULES

During the course of the development of the enhanced WCS, various modules where developed to perform useful tasks such as obtaining the current date and time. In addition, errors were found in some of the baseline WCS modules and corrected.

4.8.1 Date

The date module, DOSDATE.ASM, is an assembly language subroutine which returns the current system date. This module uses the PC-DOS interrupt function 2AH to get the current year, month, day, and day of the week in binary form.

4.8.2 Time of Day

The time module, DOSTIME.ASM is an assembly language subroutine which returns the current system time of day. This module uses the PC-DOS interrupt function 2CH to get the current time in hours, minutes, seconds, and hundredths of a second. The accuracy of this module is dependent upon the computer's timekeeping hardware. On some computers the value of hundredths of a second may only be an

approximation.

4.8.3 Timer

The timer module, BIOSTIME.ASM, is an assembly language subroutine which returns either the current system time in timer ticks past midnight or the time that has passed since the module was last called. Each timer tick is equal to one cycle of an internal system clock, approximately 55 milliseconds. This module uses an IBMBIOS interrupt function 1AH.

4.8.4 Q Function

The Q fuction module, QFUNCT.FAP, is used to evaluate the Gaussian tail function $Q(x)$[20], defined according to the equation

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2}\, dy$$

using a polynomial approximation[21]. For the case where $x \geq 0$,

$$Q(x) = \frac{1}{\sqrt{2\pi}}(b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5) e^{-x^2/2}$$

where

$$t = \frac{1}{1 + px}$$

and the constants are defined as follows

$$b_1 = +0.319381530$$

$$b_2 = -0.356563782$$

$$b_3 = +1.781477937$$

$$b_4 = -1.821255978$$

$$b_5 = +1.330274429$$

$$p = 0.2316419$$

For the case where $x < 0$, $Q(x)$ can be evaluated using

$$Q(x) = 1 - Q(-x)$$

where $Q(-x)$ is evaluated using the approximation above.

This function, although included in the baseline WCS, was implemented incorrectly. The operation of the new module has been verified using published tables of the distribution function of a zero mean unit variance Gaussian random variable which is equal to $1 - Q(x)$.

SECTION 5

ENHANCED WCS PERFORMANCE

The operation of all of the modules developed for the enhanced WCS was verified using both a standalone subset of the enhanced WCS and the enhanced WCS in it entirety.

5.1 Enhanced WCS Subset Performance

The proper operation of each of the enhanced WCS modules was first verified using a subset of the enhanced WCS. This subset was built as new modules were developed. Initially this subset consisted of only a binary source module and a binary sink module, but it grew into the system shown in the block diagram of Fig. 5.1.

The results of the simulations performed using this subset of the enhanced WCS are comparable with documented results [22,23] of simulations performed using the ICS. The results of these simulations are tabulated and plotted on the following pages for the range of values of E_s/N_0 from -10.0 dB to $+10.0$ dB for BPSK, DPSK, QPSK, and OQPSK modulation. Figures 5.2, 5.4, 5.6, 5.8 are the results for the simulations where the VORTEX/AT was not used and Figures 5.3, 5.5, 5.7, 5.9 are the results for the simulations where the VORTEX/AT was used.

The simulation run times are tabulated along with the results for each of the above simulations. The use of the VORTEX/AT array processor reduced the sim-

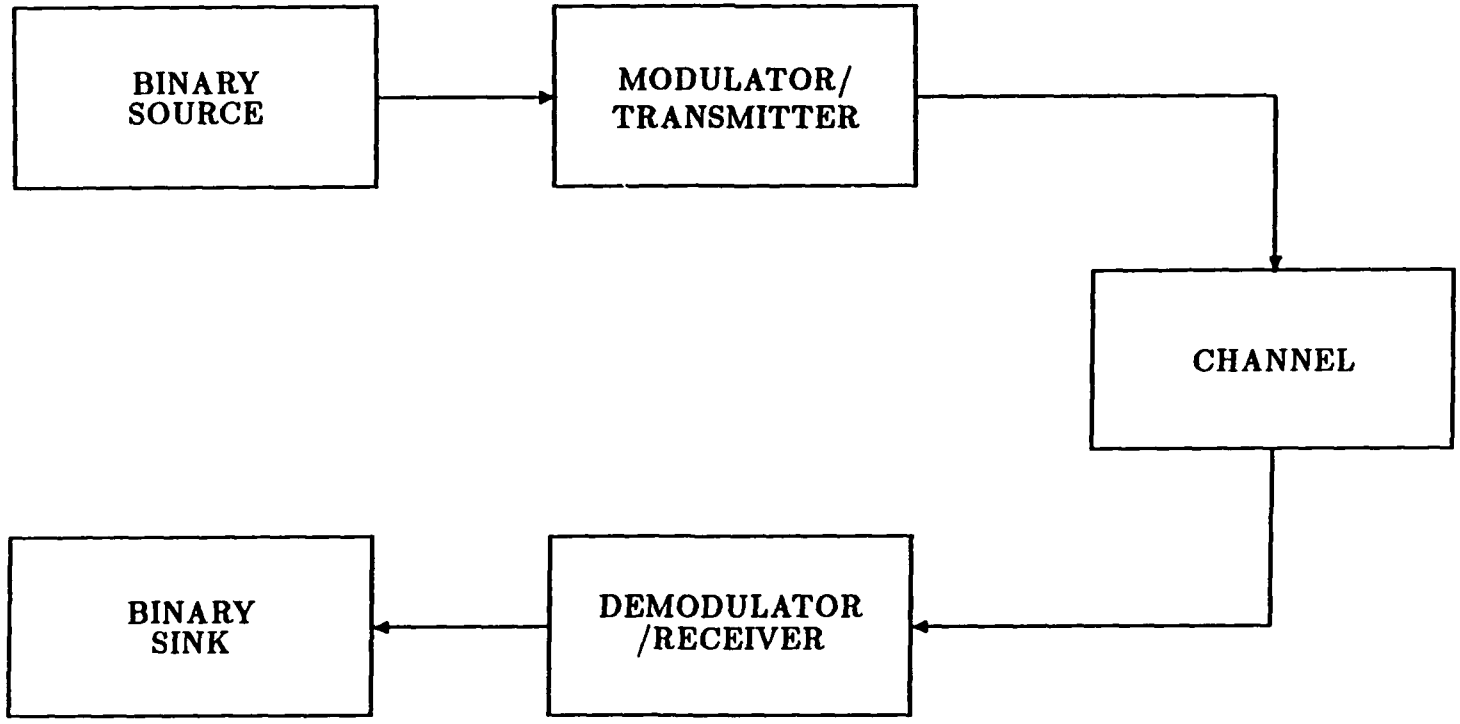

Figure 5.1: Block Diagram of Enhanced WCS Subset

ulation run times by a factor of approximately three times. While these results do not provide an exact estimate of the expected decrease in simulation times for the enhanced WCS, they do indicate the order of magnitude of the decrease expected. The factor of three decrease in run time is conservative since the VORTEX/AT microcoded Gaussian random number generator was not used in any of these simulations. In section 5.3, a more accurate estimate of the decrease in simulation times is presented.

An interesting trend can be noted in the simulation run times. As the signal–to–noise ratio increased, the simulation run times decreased. This trend can be attributed to the fact that as the signal–to–noise ratio increased, the number of errors decreased, and therefore, the time spent in counting errors decreased.

Numerical Results for BPSK, Without VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	16,232	3.25×10^{-1}	3.27×10^{-1}	38:45.44
−9.0	50,000	15,460	3.09×10^{-1}	3.08×10^{-1}	38:45.38
−8.0	50,000	14,250	2.85×10^{-1}	2.87×10^{-1}	38:45.38
−7.0	50,000	13,204	2.64×10^{-1}	2.64×10^{-1}	38:45.44
−6.0	50,000	11,866	2.37×10^{-1}	2.39×10^{-1}	38:45.16
−5.0	50,000	10,783	2.16×10^{-1}	2.13×10^{-1}	38:45.16
−4.0	50,000	9,287	1.86×10^{-1}	1.86×10^{-1}	38:45.00
−3.0	50,000	7,805	1.56×10^{-1}	1.58×10^{-1}	38:45.05
−2.0	50,000	6,489	1.30×10^{-1}	1.30×10^{-1}	38:45.06
−1.0	50,000	5,154	1.03×10^{-1}	1.04×10^{-1}	38:44.88
0.0	50,000	3,988	7.98×10^{-2}	7.86×10^{-2}	38:44.12
1.0	50,000	2,855	5.71×10^{-2}	5.63×10^{-2}	38:44.01
2.0	50,000	1,906	3.81×10^{-2}	3.75×10^{-2}	38:43.84
3.0	50,000	1,130	2.26×10^{-2}	2.29×10^{-2}	38:43.90
4.0	50,000	635	1.27×10^{-2}	1.25×10^{-2}	38:43.73
5.0	50,000	297	5.94×10^{-3}	5.95×10^{-3}	38:43.95
6.0	50,000	122	2.44×10^{-3}	2.39×10^{-3}	38:43.79
7.0	100,000	105	1.04×10^{-4}	7.73×10^{-4}	77:27.58
8.0	100,000	16	1.60×10^{-4}	1.91×10^{-4}	77:27.74
9.0	100,000	1	1.00×10^{-5}	3.37×10^{-5}	77:27.64
10.0	100,000	0	0.00×10^{-5}	3.88×10^{-5}	77:27.80

Table 5.1: Numerical Results for BPSK, without VORTEX/AT

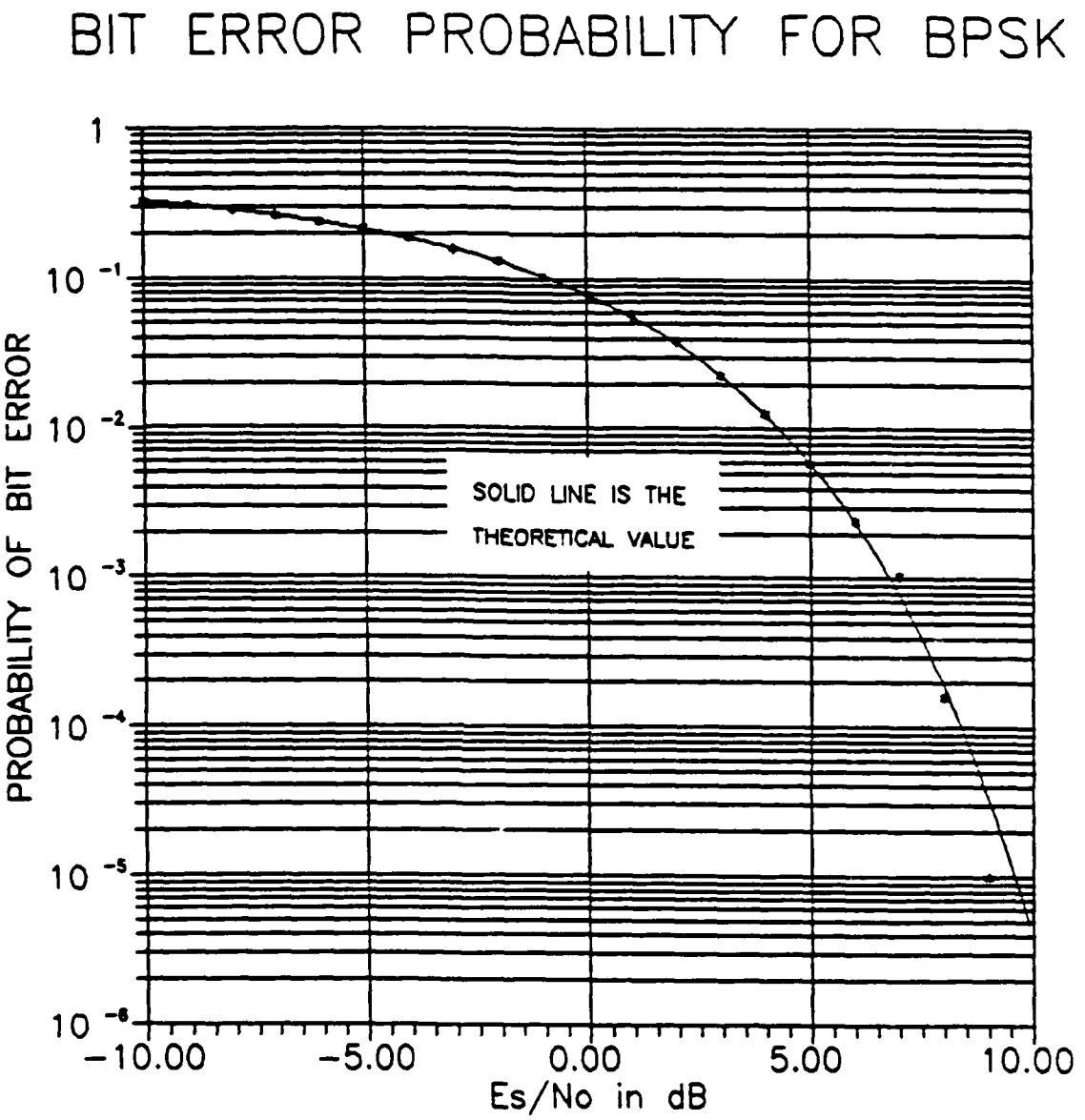

Figure 5.2: Probability of Bit Error for BPSK, without VORTEX/AT

Numerical Results for BPSK, With VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	16,409	3.28×10^{-1}	3.27×10^{-1}	10:20.88
−9.0	50,000	15,206	3.04×10^{-1}	3.08×10^{-1}	10:20.87
−8.0	50,000	14,335	2.87×10^{-1}	2.87×10^{-1}	10:20.83
−7.0	50,000	13,096	2.62×10^{-1}	2.64×10^{-1}	10:20.77
−6.0	50,000	11,888	2.38×10^{-1}	2.39×10^{-1}	10:20.71
−5.0	50,000	10,645	2.13×10^{-1}	2.13×10^{-1}	10:20.71
−4.0	50,000	9,467	1.89×10^{-1}	1.86×10^{-1}	10:20.60
−3.0	50,000	7,867	1.57×10^{-1}	1.58×10^{-1}	10:20.55
−2.0	50,000	6,651	1.33×10^{-1}	1.30×10^{-1}	10:20.49
−1.0	50,000	5,198	1.04×10^{-1}	1.04×10^{-1}	10:20.44
0.0	50,000	3,927	7.85×10^{-2}	7.86×10^{-2}	10:20.33
1.0	50,000	2,798	5.60×10^{-2}	5.63×10^{-2}	10:20.28
2.0	50,000	1,852	3.70×10^{-2}	3.75×10^{-2}	10:20.17
3.0	50,000	1,181	2.36×10^{-2}	2.29×10^{-2}	10:20.10
4.0	50,000	614	1.23×10^{-2}	1.25×10^{-2}	10:20.11
5.0	50,000	294	5.88×10^{-3}	5.95×10^{-3}	10:20.06
6.0	50,000	106	2.12×10^{-3}	2.39×10^{-3}	10:19.99
7.0	100,000	64	6.40×10^{-4}	7.73×10^{-4}	20:41.37
8.0	100,000	16	1.60×10^{-4}	1.91×10^{-4}	20:41.42
9.0	100,000	3	3.00×10^{-5}	3.37×10^{-5}	20:41.37
10.0	100,000	1	1.00×10^{-5}	3.88×10^{-5}	20:41.20

Table 5.2: Numerical Results for BPSK, with VORTEX/AT

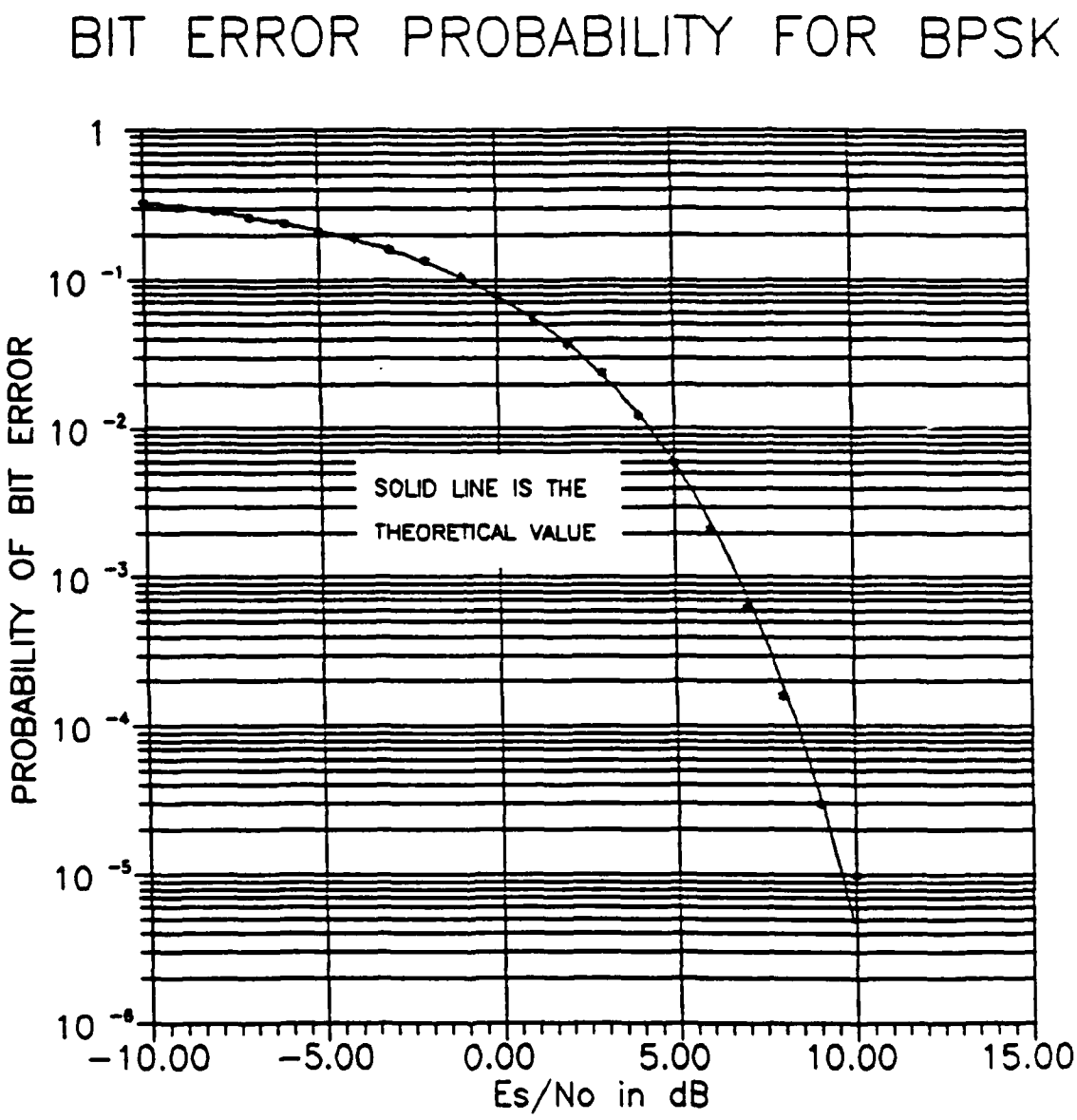

Figure 5.3: Probability of Bit Error for BPSK, with VORTEX/AT

Numerical Results for DPSK, Without VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
-10.0	50,000	22,785	4.56×10^{-1}	4.52×10^{-1}	38:56.14
-9.0	50,000	22,032	4.41×10^{-1}	4.41×10^{-1}	38:56.09
-8.0	50,000	21,307	4.26×10^{-1}	4.27×10^{-1}	38:56.15
-7.0	50,000	20,254	4.05×10^{-1}	4.10×10^{-1}	38:56.14
-6.0	50,000	19,352	3.87×10^{-1}	3.89×10^{-1}	38:56.09
-5.0	50,000	18,317	3.66×10^{-1}	3.64×10^{-1}	38:55.88
-4.0	50,000	16,785	3.36×10^{-1}	3.36×10^{-1}	38:55.87
-3.0	50,000	15,256	3.05×10^{-1}	3.03×10^{-1}	38:55.65
-2.0	50,000	13,246	2.65×10^{-1}	2.66×10^{-1}	38:55.81
-1.0	50,000	11,370	2.27×10^{-1}	2.26×10^{-1}	38:55.54
0.0	50,000	9,380	1.88×10^{-1}	1.84×10^{-1}	38:55.54
1.0	50,000	7,384	1.48×10^{-1}	1.42×10^{-1}	38:55.21
2.0	50,000	5,200	1.04×10^{-1}	1.02×10^{-1}	38:55.16
3.0	50,000	3,680	7.36×10^{-2}	6.80×10^{-2}	38:55.05
4.0	50,000	2,306	4.61×10^{-2}	4.06×10^{-2}	38:54.94
5.0	50,000	1,190	2.38×10^{-2}	2.12×10^{-2}	38:54.72
6.0	50,000	616	1.23×10^{-2}	9.33×10^{-3}	38:54.77
7.0	50,000	242	4.84×10^{-3}	3.33×10^{-3}	38:54.34
8.0	50,000	74	1.48×10^{-4}	9.09×10^{-4}	38:54.33
9.0	50,000	12	2.40×10^{-4}	1.78×10^{-4}	38:54.39
10.0	50,000	0	0.00×10^{-5}	2.27×10^{-5}	38:54.44

Table 5.3: Numerical Results for DPSK, without VORTEX/AT

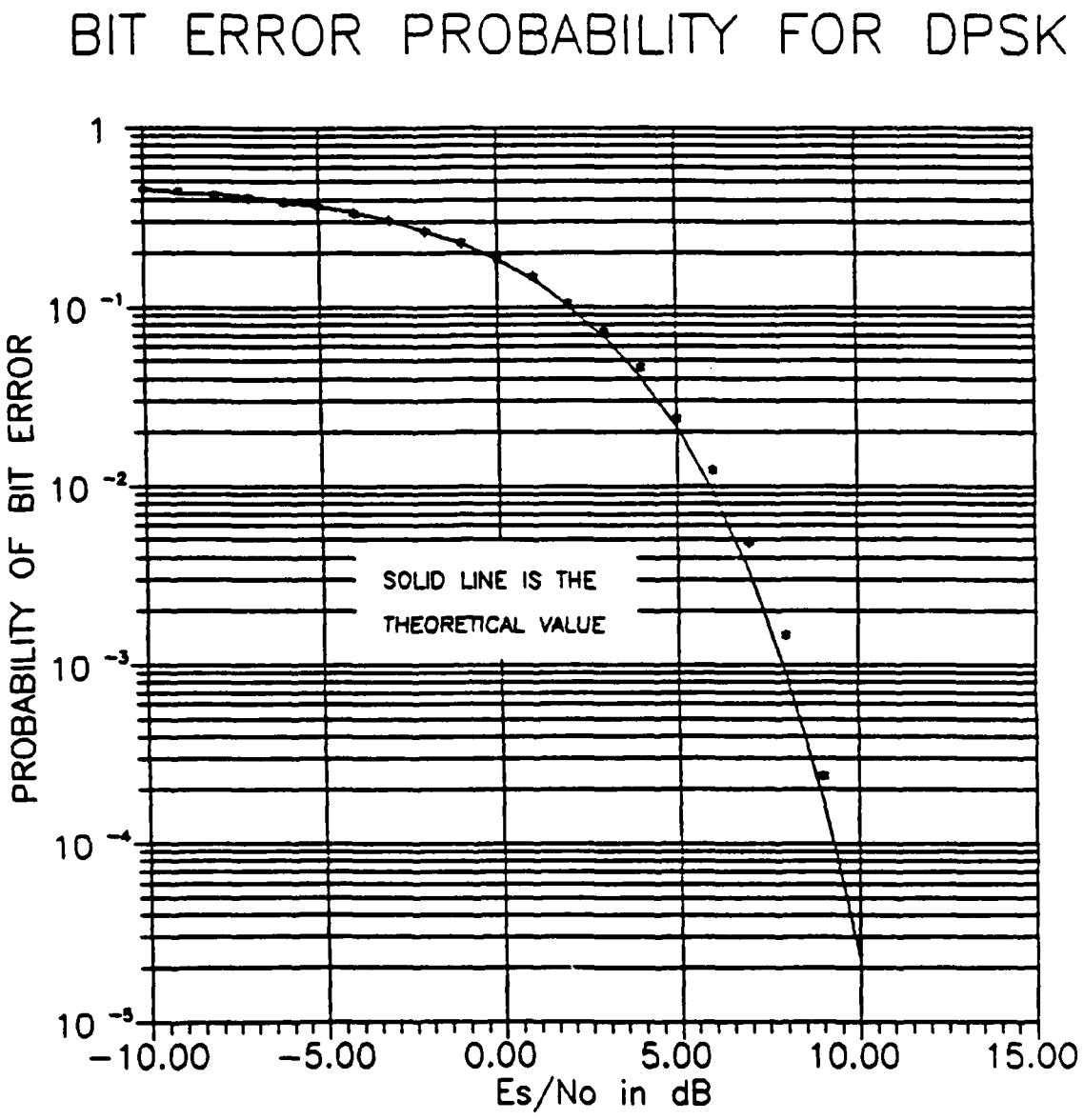

Figure 5.4: Probability of Bit Error for DPSK, without VORTEX/AT

Numerical Results for DPSK, With VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	22,787	4.56×10^{-1}	4.52×10^{-1}	10:34.94
−9.0	50,000	21,764	4.35×10^{-1}	4.41×10^{-1}	10:34.94
−8.0	50,000	21,174	4.23×10^{-1}	4.27×10^{-1}	10:34.89
−7.0	50,000	20,422	4.08×10^{-1}	4.10×10^{-1}	10:34.88
−6.0	50,000	19,380	3.88×10^{-1}	3.89×10^{-1}	10:34.83
−5.0	50,000	18,174	3.63×10^{-1}	3.64×10^{-1}	10:34.77
−4.0	50,000	16,582	3.32×10^{-1}	3.36×10^{-1}	10:34.72
−3.0	50,000	14,948	2.99×10^{-1}	3.03×10^{-1}	10:34.66
−2.0	50,000	13,076	2.62×10^{-1}	2.66×10^{-1}	10:34.56
−1.0	50,000	11,328	2.27×10^{-1}	2.26×10^{-1}	10:34.50
0.0	50,000	9,210	1.84×10^{-1}	1.84×10^{-1}	10:34.34
1.0	50,000	7,086	1.42×10^{-1}	1.42×10^{-1}	10:34.28
2.0	50,000	5,158	1.03×10^{-1}	1.02×10^{-1}	10:34.11
3.0	50,000	3,566	7.13×10^{-2}	6.80×10^{-2}	10:34.06
4.0	50,000	2,184	4.37×10^{-2}	4.06×10^{-2}	10:33.95
5.0	50,000	1,222	2.44×10^{-2}	2.12×10^{-2}	10:33.90
6.0	50,000	594	1.18×10^{-2}	9.33×10^{-3}	10:33.84
7.0	50,000	268	5.46×10^{-3}	3.33×10^{-3}	10:33.79
8.0	50,000	78	1.56×10^{-3}	9.09×10^{-4}	10:33.78
9.0	50,000	24	4.80×10^{-4}	1.78×10^{-4}	10:33.67
10.0	50,000	0	0.00×10^{-5}	2.27×10^{-5}	10:33.62

Table 5.4: Numerical Results for DPSK, with VORTEX/AT

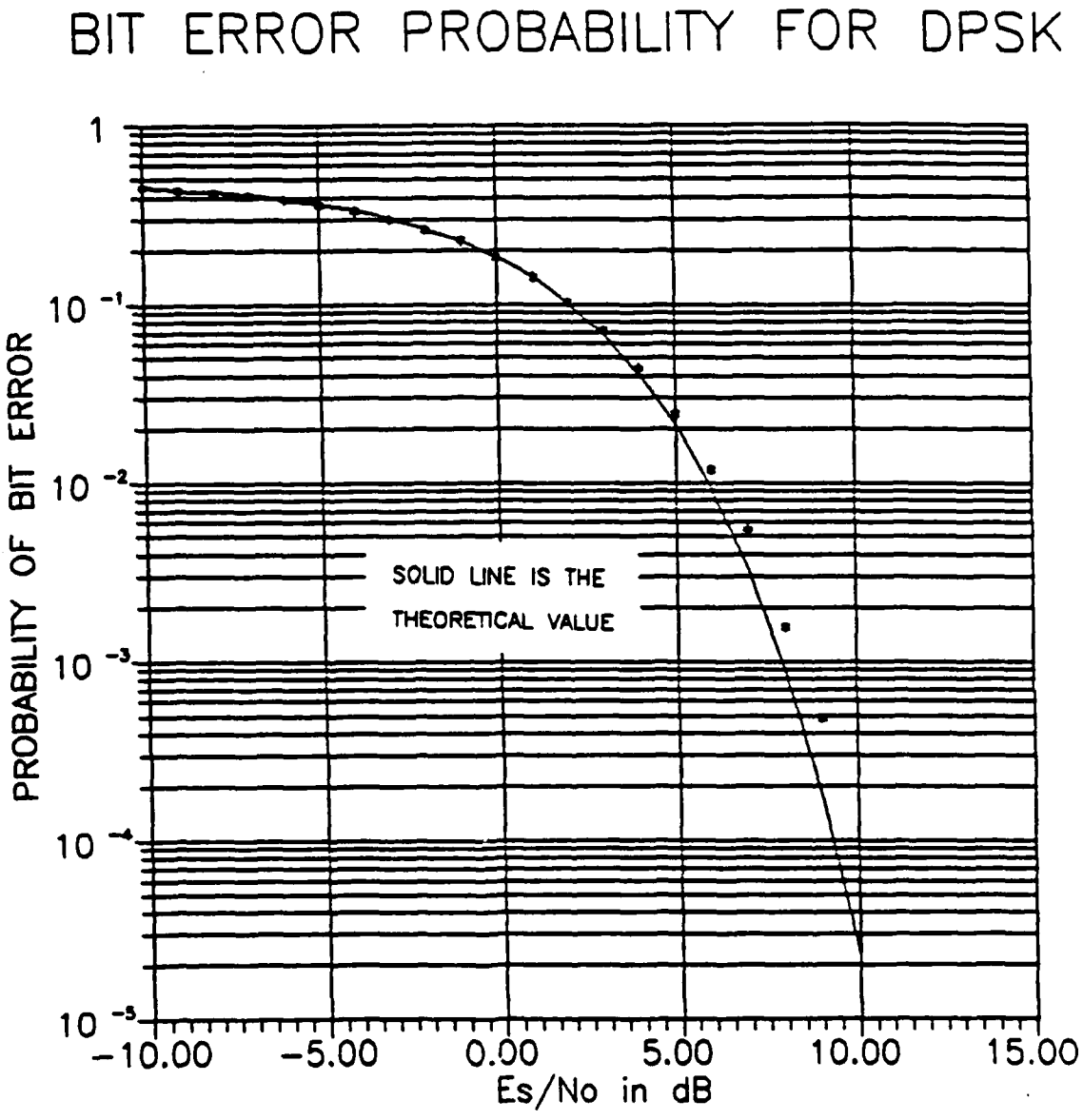

Figure 5.5: Probability of Bit Error for DPSK, with VORTEX/AT

Numerical Results for QPSK, Without VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	18,881	3.78×10^{-1}	3.76×10^{-1}	21:01.30
−9.0	50,000	18,135	3.63×10^{-1}	3.61×10^{-1}	21:01.25
−8.0	50,000	17,316	3.46×10^{-1}	3.45×10^{-1}	21:01.14
−7.0	50,000	16,416	3.28×10^{-1}	3.28×10^{-1}	21:01.14
−6.0	50,000	15,457	3.09×10^{-1}	3.08×10^{-1}	21:01.09
−5.0	50,000	14,378	2.88×10^{-1}	2.87×10^{-1}	21:01.09
−4.0	50,000	13,272	2.65×10^{-1}	2.64×10^{-1}	21:01.03
−3.0	50,000	12,000	2.40×10^{-1}	2.39×10^{-1}	21:00.82
−2.0	50,000	10,722	2.14×10^{-1}	2.14×10^{-1}	21:00.82
−1.0	50,000	9,402	1.88×10^{-1}	1.86×10^{-1}	21:00.70
0.0	50,000	7,982	1.60×10^{-1}	1.59×10^{-1}	21:00.70
1.0	50,000	6,557	1.31×10^{-1}	1.31×10^{-1}	21:00.65
2.0	50,000	5,192	1.04×10^{-1}	1.04×10^{-1}	21:00.60
3.0	50,000	3,905	7.81×10^{-2}	7.89×10^{-2}	21:00.54
4.0	50,000	2,764	5.53×10^{-2}	5.65×10^{-2}	21:00.16
5.0	50,000	1,824	3.65×10^{-2}	3.77×10^{-2}	21:00.10
6.0	50,000	1,103	2.21×10^{-2}	2.30×10^{-2}	21:00.10
7.0	50,000	588	1.18×10^{-2}	1.26×10^{-2}	21:00.05
8.0	100,000	554	5.54×10^{-3}	6.00×10^{-3}	42:03.66
9.0	100,000	224	2.24×10^{-3}	2.41×10^{-3}	42:03.66
10.0	100,000	75	7.50×10^{-4}	7.83×10^{-4}	42:02.89

Table 5.5: Numerical Results for QPSK, without VORTEX/AT

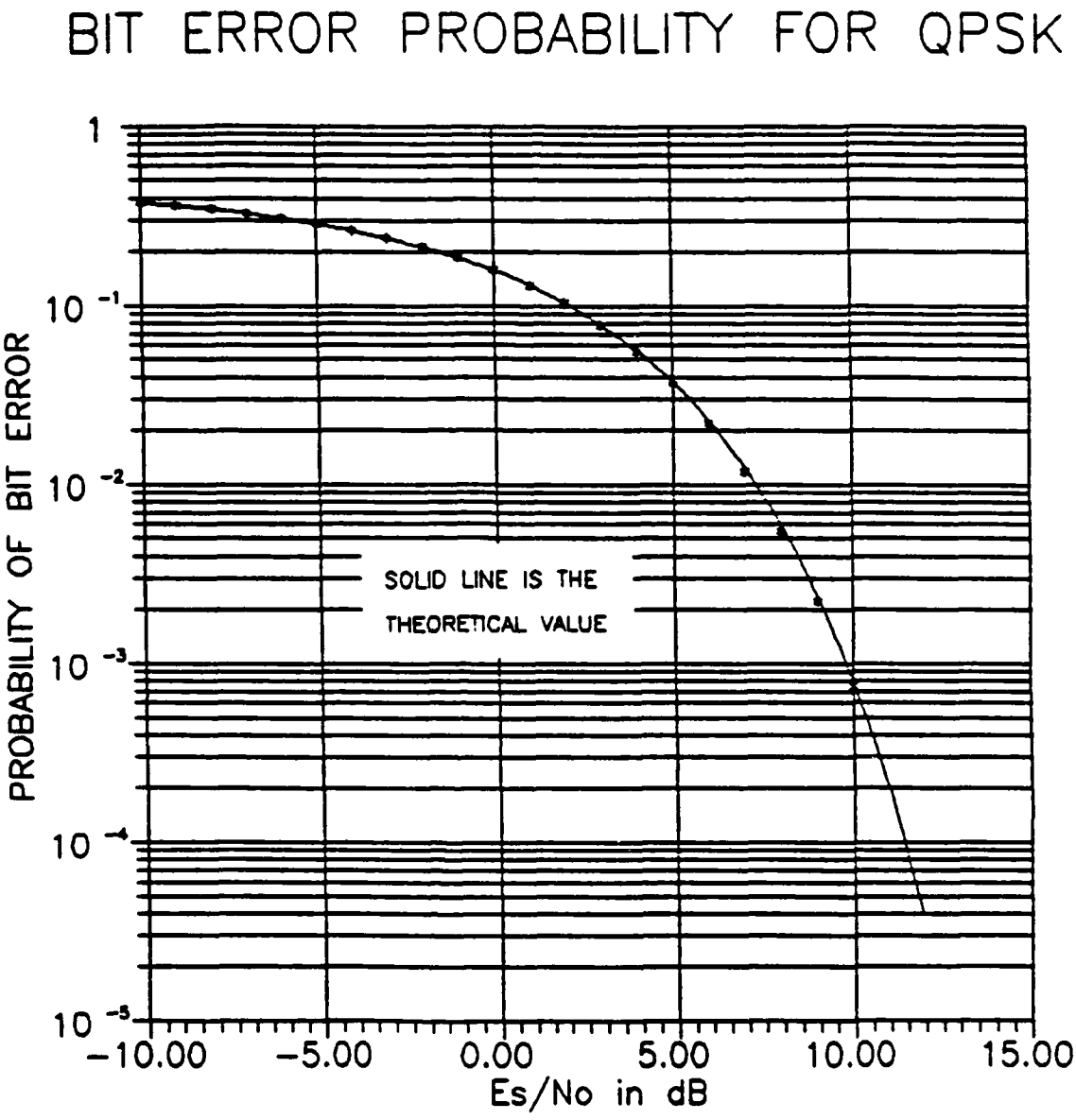

Figure 5.6: Probability of Bit Error for QPSK, without VORTEX/AT

Numerical Results for QPSK, With VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	18,784	3.76×10^{-1}	3.76×10^{-1}	7:56.04
−9.0	50,000	18,040	3.61×10^{-1}	3.61×10^{-1}	7:55.99
−8.0	50,000	17,241	3.45×10^{-1}	3.45×10^{-1}	7:55.93
−7.0	50,000	16,312	3.26×10^{-1}	3.28×10^{-1}	7:55.99
−6.0	50,000	15,326	3.07×10^{-1}	3.08×10^{-1}	7:55.88
−5.0	50,000	14,337	2.87×10^{-1}	2.87×10^{-1}	7:55.88
−4.0	50,000	13,184	2.64×10^{-1}	2.64×10^{-1}	7:55.82
−3.0	50,000	11,936	2.39×10^{-1}	2.39×10^{-1}	7:55.77
−2.0	50,000	10,590	2.12×10^{-1}	2.14×10^{-1}	7:55.71
−1.0	50,000	9,195	1.84×10^{-1}	1.86×10^{-1}	7:55.60
0.0	50,000	7,819	1.56×10^{-1}	1.59×10^{-1}	7:55.60
1.0	50,000	6,502	1.30×10^{-1}	1.31×10^{-1}	7:55.38
2.0	50,000	5,144	1.03×10^{-1}	1.04×10^{-1}	7:55.32
3.0	50,000	3,940	7.88×10^{-2}	7.89×10^{-2}	7:55.38
4.0	50,000	2,797	5.59×10^{-2}	5.65×10^{-2}	7:55.33
5.0	50,000	1,881	3.76×10^{-2}	3.77×10^{-2}	7:55.27
6.0	50,000	1,149	2.30×10^{-2}	2.30×10^{-2}	7:55.21
7.0	50,000	619	1.24×10^{-2}	1.26×10^{-2}	7:55.16
8.0	100,000	593	5.93×10^{-3}	6.00×10^{-3}	15:50.04
9.0	100,000	227	2.27×10^{-3}	2.41×10^{-3}	15:49.93
10.0	100,000	72	7.20×10^{-4}	7.83×10^{-4}	15:49.88

Table 5.6: Numerical Results for QPSK, with VORTEX/AT

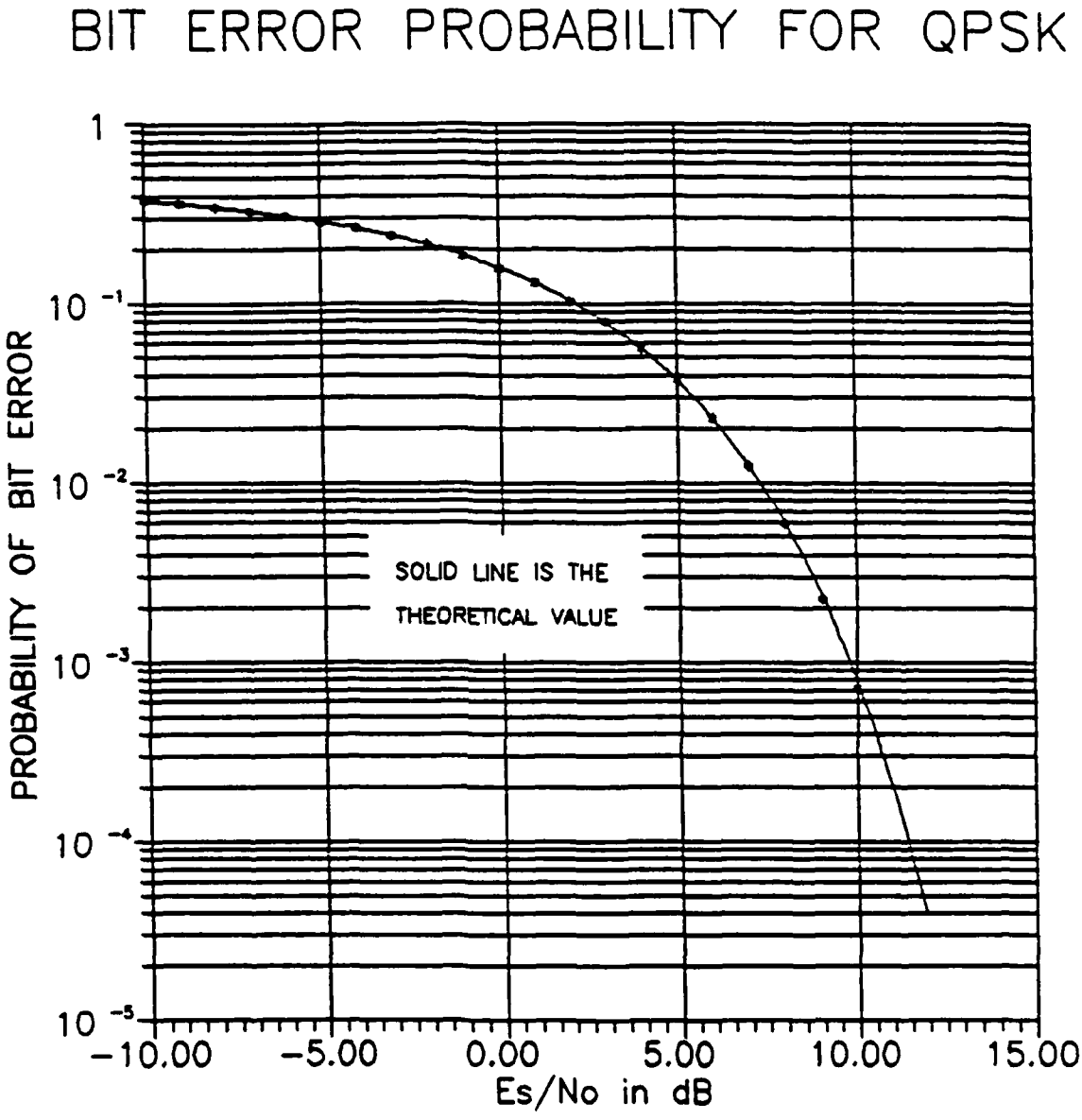

Figure 5.7: Probability of Bit Error for QPSK, with VORTEX/AT

Numerical Results for OQPSK, Without VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
-10.0	50,000	18,856	3.77×10^{-1}	3.76×10^{-1}	21:36.26
-9.0	50,000	18,148	3.63×10^{-1}	3.61×10^{-1}	21:36.18
-8.0	50,000	17,319	3.46×10^{-1}	3.45×10^{-1}	21:36.13
-7.0	50,000	16,405	3.28×10^{-1}	3.28×10^{-1}	21:36.08
-6.0	50,000	15,446	3.09×10^{-1}	3.08×10^{-1}	21:36.03
-5.0	50,000	14,373	2.87×10^{-1}	2.87×10^{-1}	21:36.08
-4.0	50,000	13,235	2.65×10^{-1}	2.64×10^{-1}	21:36.02
-3.0	50,000	12,013	2.40×10^{-1}	2.39×10^{-1}	21:35.80
-2.0	50,000	10,741	2.15×10^{-1}	2.14×10^{-1}	21:35.74
-1.0	50,000	9,405	1.88×10^{-1}	1.86×10^{-1}	21:35.64
0.0	50,000	7,938	1.59×10^{-1}	1.59×10^{-1}	21:35.70
1.0	50,000	6,517	1.30×10^{-1}	1.31×10^{-1}	21:35.58
2.0	50,000	5,155	1.03×10^{-1}	1.04×10^{-1}	21:35.53
3.0	50,000	3,889	7.78×10^{-2}	7.89×10^{-2}	21:35.48
4.0	50,000	2,762	5.52×10^{-2}	5.65×10^{-2}	21:35.15
5.0	50,000	1,814	3.63×10^{-2}	3.77×10^{-2}	21:35.09
6.0	50,000	1,104	2.21×10^{-2}	2.30×10^{-2}	21:35.09
7.0	50,000	562	1.12×10^{-2}	1.26×10^{-2}	21:35.04
8.0	100,000	598	5.98×10^{-3}	6.00×10^{-3}	43:14.79
9.0	100,000	261	2.61×10^{-3}	2.41×10^{-3}	43:14.73
10.0	100,000	98	9.80×10^{-4}	7.83×10^{-4}	43:14.02

Table 5.7: Numerical Results for OQPSK, without VORTEX/AT

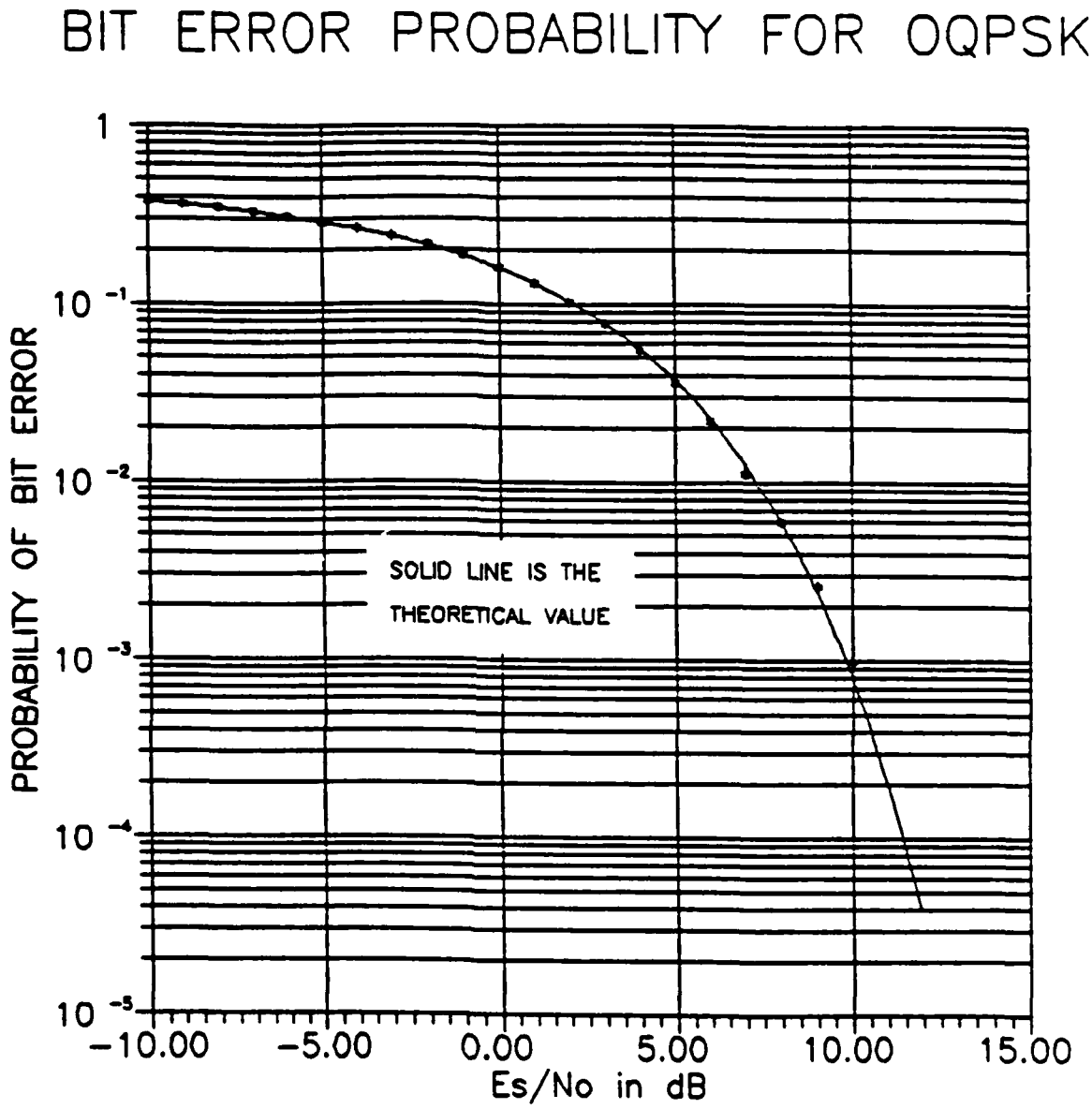

Figure 5.8: Probability of Bit Error for OQPSK, without VORTEX/AT

Numerical Results for OQPSK, With VORTEX/AT					
E_s/N_0 (dB)	Number of Source Bits	Number of Bit Errors	Experimental P_e	Theoretical P_e	Run Time (min:sec)
−10.0	50,000	18,760	3.75×10^{-1}	3.76×10^{-1}	8:31.74
−9.0	50,000	17,986	3.60×10^{-1}	3.61×10^{-1}	8:31.74
−8.0	50,000	17,138	3.43×10^{-1}	3.45×10^{-1}	8:31.58
−7.0	50,000	16,226	3.25×10^{-1}	3.28×10^{-1}	8:31.69
−6.0	50,000	15,342	3.07×10^{-1}	3.08×10^{-1}	8:31.63
−5.0	50,000	14,276	2.86×10^{-1}	2.87×10^{-1}	8:31.63
−4.0	50,000	13,159	2.63×10^{-1}	2.64×10^{-1}	8:31.53
−3.0	50,000	11,941	2.39×10^{-1}	2.39×10^{-1}	8:31.53
−2.0	50,000	10,577	2.12×10^{-1}	2.14×10^{-1}	8:31.41
−1.0	50,000	9,221	1.84×10^{-1}	1.86×10^{-1}	8:31.25
0.0	50,000	7,869	1.57×10^{-1}	1.59×10^{-1}	8:31.30
1.0	50,000	6,485	1.30×10^{-1}	1.31×10^{-1}	8:31.25
2.0	50,000	5,129	1.03×10^{-1}	1.04×10^{-1}	8:31.19
3.0	50,000	3,912	7.82×10^{-2}	7.89×10^{-2}	8:31.14
4.0	50,000	2,786	5.57×10^{-2}	5.65×10^{-2}	8:31.09
5.0	50,000	1,872	3.74×10^{-2}	3.77×10^{-2}	8:31.03
6.0	50,000	1,162	2.32×10^{-2}	2.30×10^{-2}	8:30.97
7.0	50,000	637	1.27×10^{-2}	1.26×10^{-2}	8:30.92
8.0	100,000	630	6.30×10^{-3}	6.00×10^{-3}	16:59.97
9.0	100,000	256	2.56×10^{-3}	2.41×10^{-3}	16:59.91
10.0	100,000	88	8.80×10^{-4}	7.83×10^{-4}	16:59.85

Table 5.8: Numerical Results for OQPSK, with VORTEX/AT

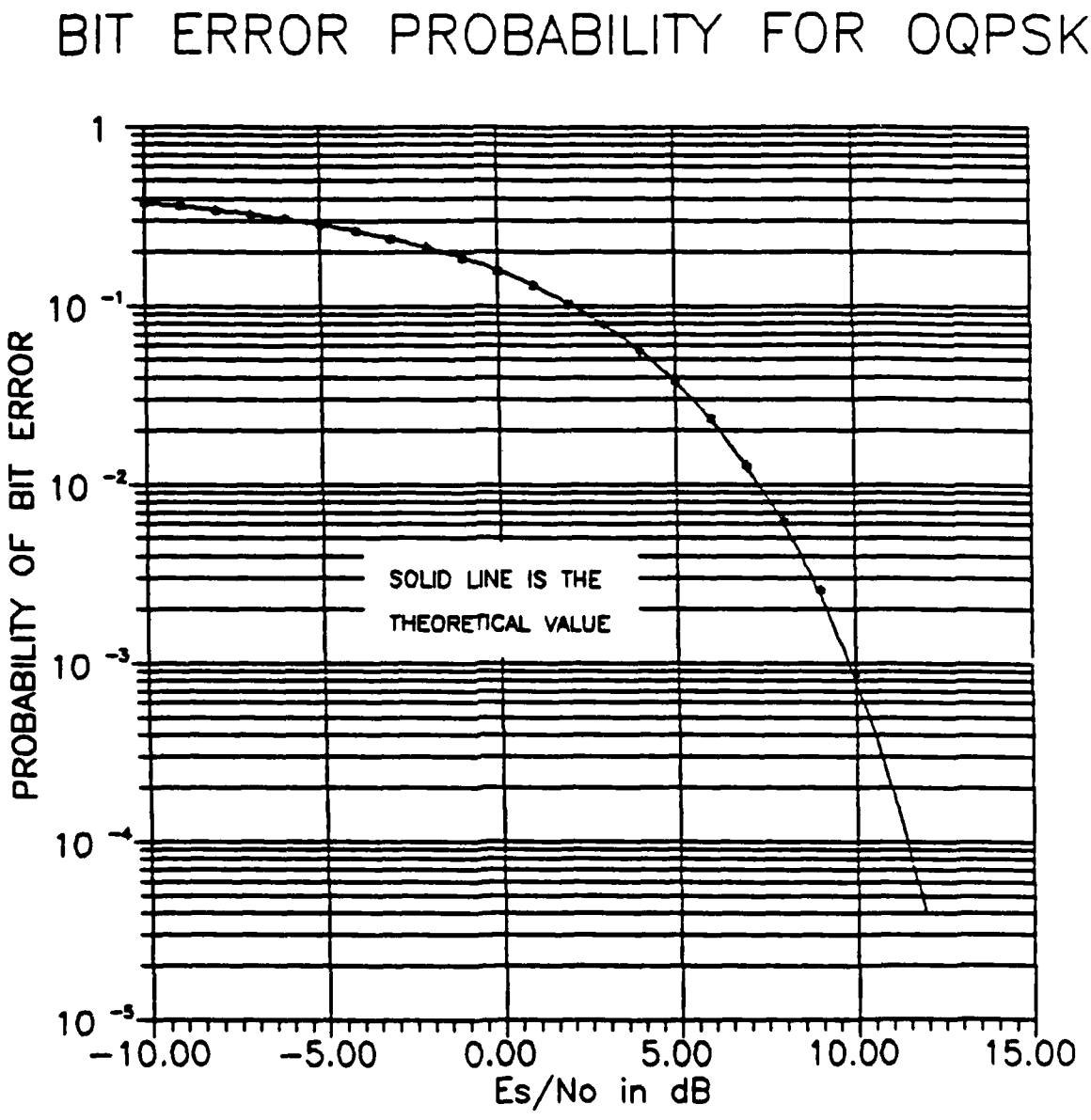

Figure 5.9: Probability of Bit Error for OQPSK, with VORTEX/AT

5.2 Enhanced WCS Performance

The proper operation of each of the enhanced WCS modules was also verified through their substitution into the baseline WCS. The VORTEX/AT was not used in any of these simulations due to the fact that it was not available. The lack of the VORTEX/AT during these simulations was not considered serious since the proper operation of the modules with the VORTEX/AT was verified using the enhanced WCS subset simulations. The purpose of these simulations was to verify that the newly developed modules were compatible with the existing WCS modules.

The numerical results of the simulations performed using the entire WCS are comparable with the results obtained using the enhanced WCS subset. The results of the simulations are plotted on the following pages for the range of values of signal-to-noise ratio (E_b/N_0) from 0.0 dB to +10.0 dB for BPSK, DPSK, QPSK, and OQPSK modulation. Also shown on the plots are curves representing the expected theoretical results. These plots were all obtained using the graphical capabilities of the WCS.

The plots of simulation results obtained using the entire WCS exhibit more scatter around the theoretical curves than those of simulation results obtained using the WCS subset. This apparent difference in the accuracy of the simulation results is due to the fact that shorter simulations were run when the entire WCS was used, rather than to any errors in the WCS itself.

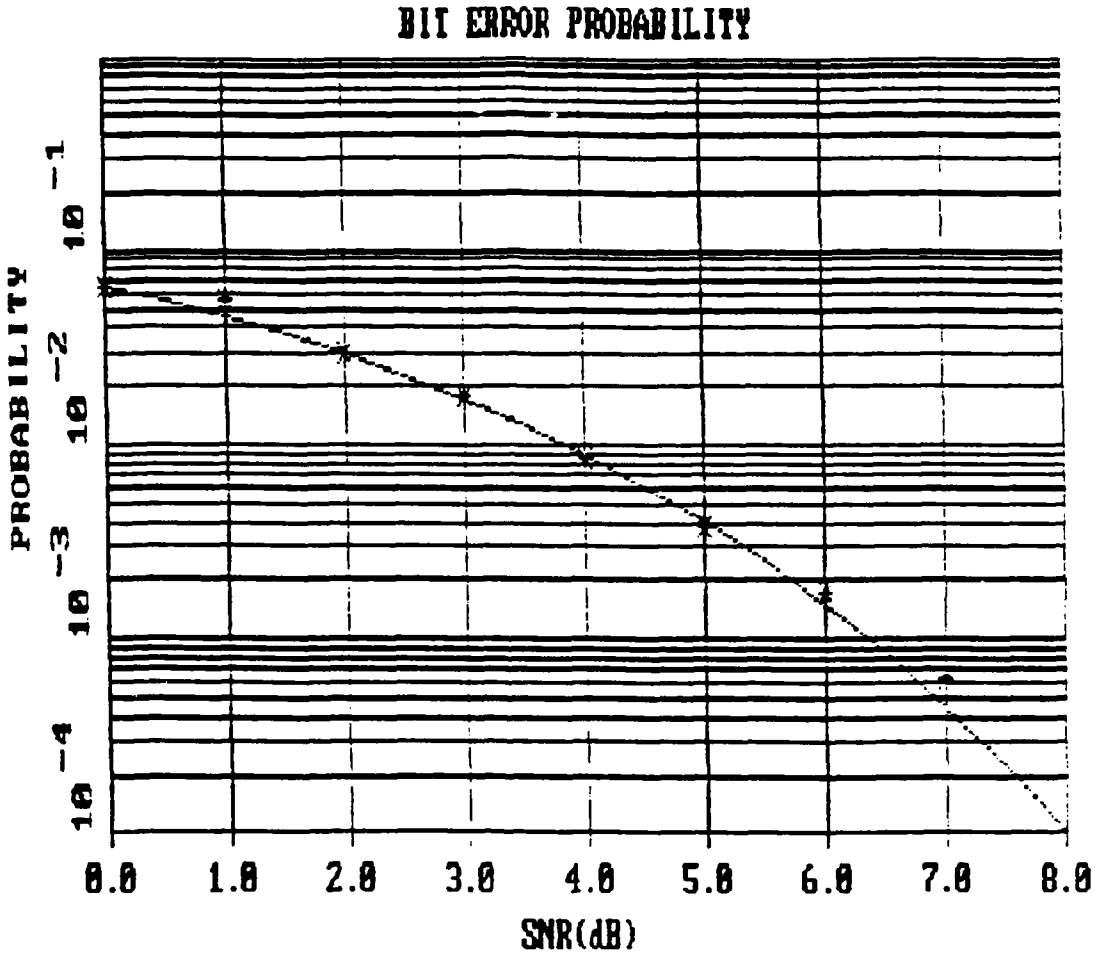

Figure 5.10: Probability of Bit Error for BPSK

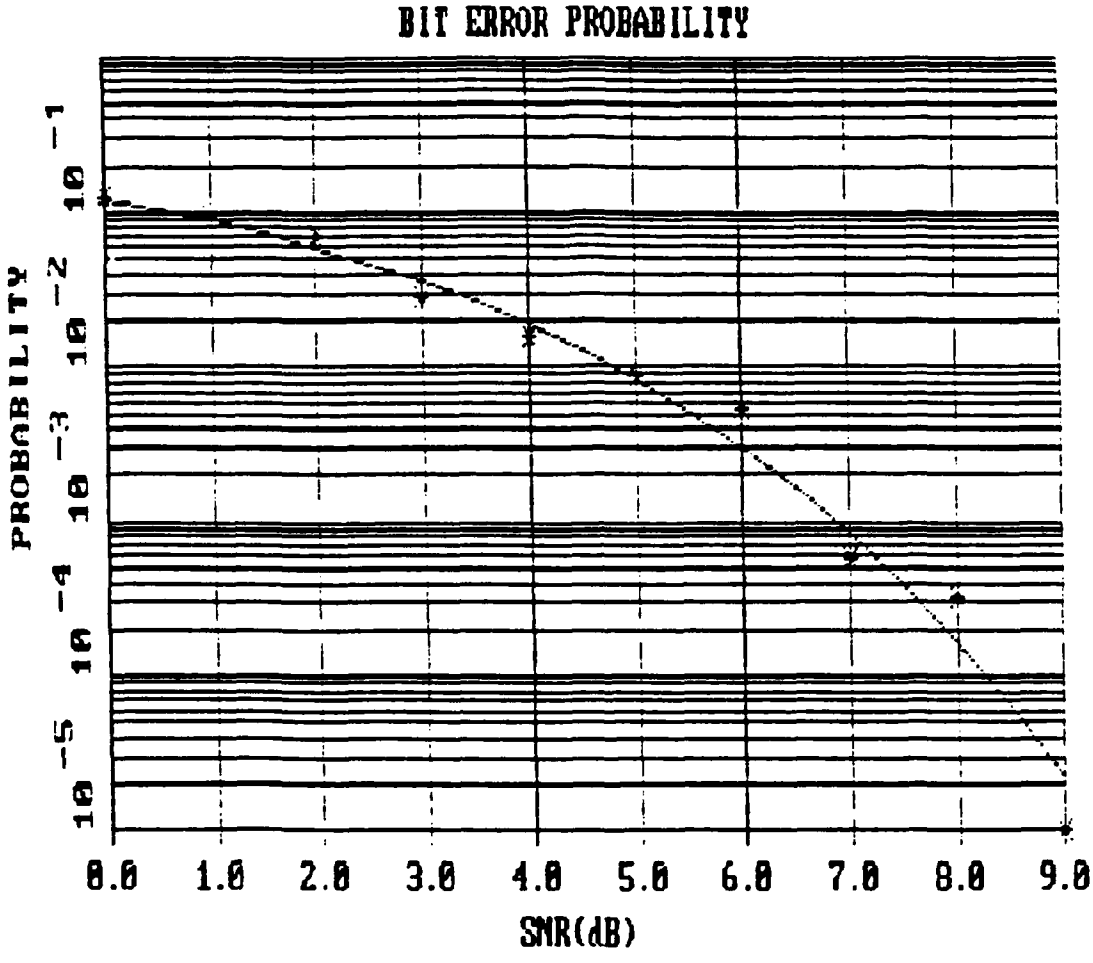

Figure 5.11: Probability of Bit Error for DPSK

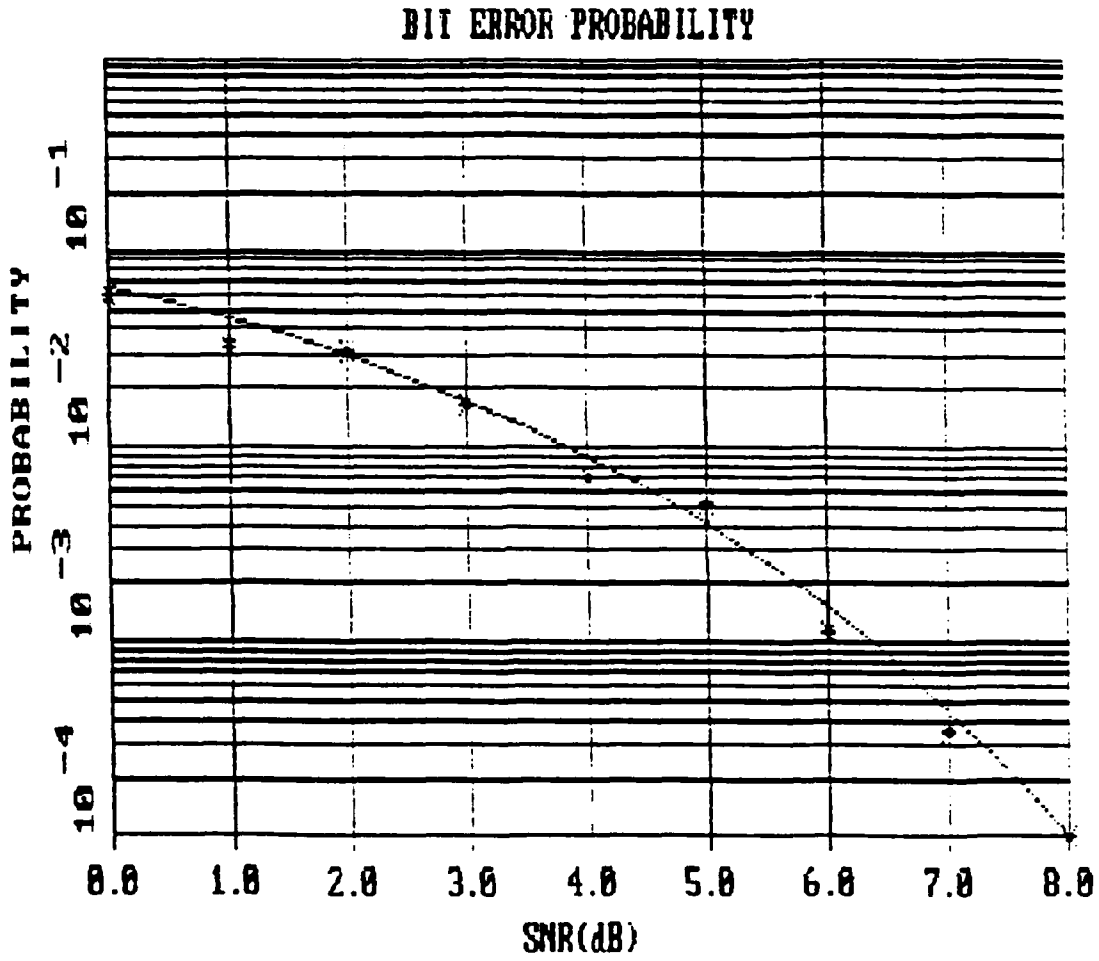

Figure 5.12: Probability of Bit Error for QPSK

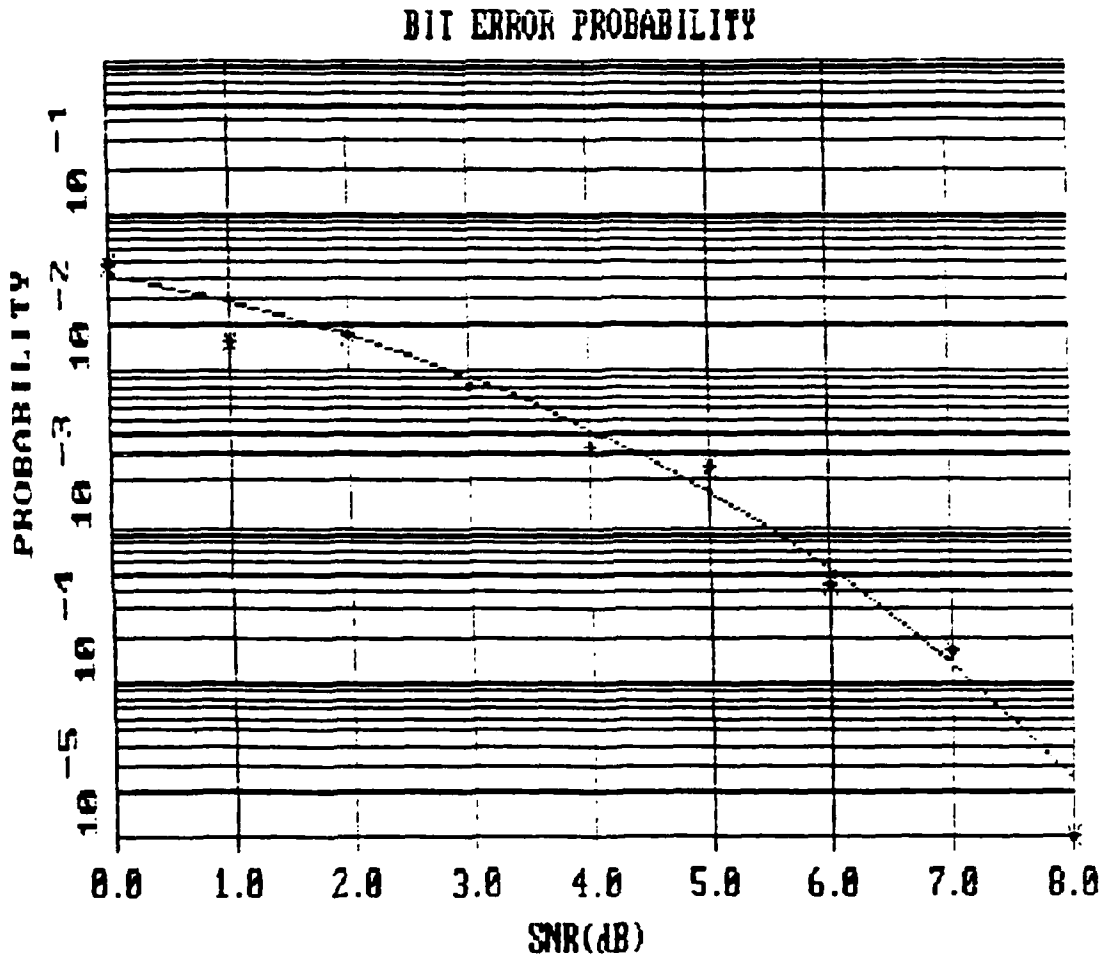

Figure 5.13: Probability of Bit Error for OQPSK

5.3 Enhanced WCS Timing Estimate

The increase in the computational throughput of the enhanced WCS possible when the VORTEX/AT was used could only be estimated. The major reason why actual timing calculations could not be made was the fact that the VORTEX/AT microcoded Gaussian random number generator did not function properly and was not used in the enhanced WCS subset simulations.

In order to generate timing estimates, the simulations using the enhanced WCS subset were rerun for BPSK modulation using the VORTEX/AT microcoded Gaussian random number generator. While the results of these simulations were not valid, the run times were. Two series of simulations were run, both consisting of 50,000 input data bits. In the first, the processing block length was 1600 bits, the same as in the enhanced WCS subset simulations. In the second series of simulations, the processing block length was 1024 bits, the same as the default value used in the enhanced WCS. The results of these simulations are tabulated in Tables 5.9 and 5.10.

The results indicate that a computational throughput gain of approximately 5.5 times can be expected in the enhanced WCS when the VORTEX/AT is used. The results also indicate that if the default processing block length in the enhanced WCS is increased, the computational throughput will be further increased.

Simulation Run Time (min:sec)			
E_b/N_0 (dB)	Baseline WCS	Enhanced WCS	VORTEX/AT Microcode
−10	38 : 45.44	10 : 20.88	6 : 44.97
−5	38 : 45.16	10 : 20.71	6 : 45.24
0	38 : 45.16	10 : 20.33	6 : 45.29
5	38 : 43.95	10 : 20.06	6 : 45.30
10	38 : 45.51	10 : 19.83	6 : 45.24

Table 5.9: Simulation Times, 50,000 Input Data Bits, 1600 Bit Processing Blocks

Simulation Run Time (min:sec)			
E_b/N_0 (dB)	Baseline WCS	Enhanced WCS	VORTEX/AT Microcode
−10	38 : 45.16	12 : 30.83	7 : 04.63
−5	38 : 44.89	12 : 30.72	7 : 05.17
0	38 : 44.29	12 : 30.67	. 7 : 04.96
5	38 : 44.01	12 : 30.23	7 : 05.04
10	38 : 43.68	12 : 30.00	7 : 05.02

Table 5.10: Simulation Times, 50,000 Input Data Bits, 1024 Bit Processing Blocks

SECTION 6

DISCUSSION AND CONCLUSIONS

This project dealt with the integration of the VORTEX/AT array processor into the Workstation Communications Simulator. The emphasis of this effort was the demonstration of the feasability of incorporating the VORTEX/AT into the WCS rather than maximizing its performance. Nonetheless, the use of the VORTEX/AT increased the throughput of the WCS.

The computational throughput gain actually obtained was consistent with expectations but lower than considered possible. This non-optimum performance was due to the fact that the Gaussian random numbers were generated using transformations of uniform random numbers rather than using the VORTEX/AT microcoded Gaussian random number generator. The Gaussian random number generator was not used because it did not generate random numbers with a standard normal distribution. The throughput gain possible when the microcoded Gaussian random number generator is used was estimated to be on the order of a factor of five to six times.

The VORTEX/AT array processor proved to be fairly easy to install and program, even without the aid of the VEX preprocessor. Several minor errors in the VORTEX/AT software were discovered which required the VORTEX/AT microcode to be downloaded before each program run, but this did not present any major difficulties. The manufacturer, Sky Computers, claimed that all of these errors will be

corrected in the next release of the VORTEX/AT software.

The work performed under this effort provides a starting point for significant extensions in many directions. These extensions would more fully utilize the capabilities of the VORTEX/AT.

First, a considerable further increase in throughput could be obtained if all of the data used in a simulation were located in the VORTEX/AT data memory and all calculations where performed in the VORTEX/AT. This would eliminate the overhead involved in passing data to and from the VORTEX/AT each time it performs a calculation. This mode of operation has been planned for through the passing of the common block APARY to the VORTEX/AT in the module APINIT.FAP. However, until an operating system which can access more than 640KB of memory is available for the IBM PC/AT, this extension is not possible.

Second, only a fraction of the WCS modules have been rewritten to utilize the capabilities of the VORTEX/AT. For example, the modulator/transmitter modules and demodulator/receiver modules, while extensively rewritten, do not currently use the VORTEX/AT. These modules should again be rewritten to utilize the VORTEX/AT when the VEX preprocessor becomes available.

Third, the entire structure of the WCS should be reexamined to ensure that it is optimum for vector operations. The WCS modules should all use vector operations rather than operations on individual numbers to the maximum extent possible. In addition, the vectors should all be as large as possible in order to minimize

the percentage of time spent in setting up a calculation relative to performing the calculation. As an example, the original version of the uniform random number generator module, RNG1R.FAP, generated only a single random number each time it was called, rather than generating all of the random numbers required for an operation.

Fourth, the conversion of some WCS modules into VORTEX/AT microcode should be examined. This may result in a further increase in simulation throughput.

The above suggestions are examples of only a few of the many extensions possible to the enhance WCS to more fully utilize the capabilities of the VORTEX/AT. As familiarity with the VORTEX/AT increases, many other additions and changes to the enhanced WCS modules will become apparent.

LITERATURE CITED

[1] James W. Modestino, Kurt R. Matis, K. Y. Jung, and Acie L. Vickers. *Design and Implementation of the Interactive Communications Simulator (ICS).* Final Technical Report RADC-TR-81-37, Rome Air Development Center, Griffiss AFB, NY 13441, April 1981.

[2] Kurt R. Matis. *Interactive Communications Simulator (ICS).* Final Technical Report RADC-TR-84-227, Rome Air Development Center, Griffiss AFB, NY 13441, November 1984.

[3] Kurt R. Matis and James W. Modestino. Interactive communication system simulation on a high-speed pc workstation. *IEEE Journal on Selected Areas in Communications*, 6(1):24–33, January 1988. Special Issue on Computer-Aided Modeling, Analysis, and Design of Communication Systems II.

[4] Tom Manuel. Vector processing comes to the desktop. *Electronics*, 60(5):69–70, March 5 1987.

[5] K. Y. Jung. *Digital Communication in the Presence of Impulsive Noise.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY 12181, December 1982. For May 1983 Graduation.

[6] John L. Ramsey. Realization of optimum interleavers. *IEEE Transactions on Information Theory*, IT-16(3):338–345, May 1970.

[7] Phillip A. Bello. Characterization of randomly time-variant linear channels. *IEEE Transactions on Communications*, CS-11(4):360–393, December 1963.

[8] R. S. Kennedy. *Fading Dispersive Communication Channels.* Wiley-Interscience, 1969.

[9] Andrew J. Viterbi. Convolutional codes and their performance in communication systems. *IEEE Transactions on Communication Technology*, COM-19(5):751–772, October 1973.

[10] G. David Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[11] J. K. Wolf. Efficient maximum likelihood decoding of linear block codes using a trellis. *IEEE Transactions on Information Theory*, IT-24(1):76–81, January 1978.

[12] Supercomputing on a pc. *Computer-Aided Engineering*, 6(4):9, April 1987.

[13] *VORTEX FORTRAN Programmer's Reference Manual.* Sky Computers, Inc., Foot of John St, Lowell MA 01852, 1.0 edition, October 1986.

[14] B. A. Wichmann and I. D. Hill. An efficient and portable pseudo-random number generator. *Applied Statistics*, 31(2):188–190, 1982.

[15] Rodney F. W. Coates, Gareth J. Janacek, and Kenneth V. Lever. Monte carlo simulation and random number generation. *IEEE Journal on Selected Areas in Communications*, 6(1):58–66, January 1988. Special Issue on Computer-Aided Modeling, Analysis, and Design of Communications Systems II.

[16] M. B. Wilk and R. Gnanadesikan. Probability plotting methods for the analysis of data. *Biometrika*, 55(1):1–17, January 1968.

[17] G. E. P. Box and Mervin E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29():610–611, 1958.

[18] James W. Modestino. *Incorporation of DPSK Modulation Formats into Communications Simulator.* Technical Report TM77-5, Rensselaer Polytechnic Institute, December 1977. Unpublished RPI Report.

[19] James W. Modestino. *Channel Signal Scaling in the Communications Simulator.* Technical Report TM77-1, Rensselaer Polytechnic Institute, September 1977. Unpublished RPI Report.

[20] James W. Modestino. *Evaluation of the Gaussian Tail Function.* Technical Report TM78-10, Rensselaer Polytechnic Institute, March 1978. Unpublished RPI Report.

[21] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* National Bureau of Standards, 1972.

[22] David J. Hayward. *Independence of Communications Simulator to the Number of Samples per Baud.* Technical Report TM78-14, Rensselaer Polytechnic Institute, April 1978. Unpublished RPI Report.

[23] Reuben Chang. *Numerical Results for: DPSK, QPSK, Offset QPSK, and BFSK Modulation Formats in Communications Simulator.* Technical Report TM78-24, Rensselaer Polytechnic Institute, August 1978. Unpublished RPI Report.